Introduksjon til Flux

Hva?

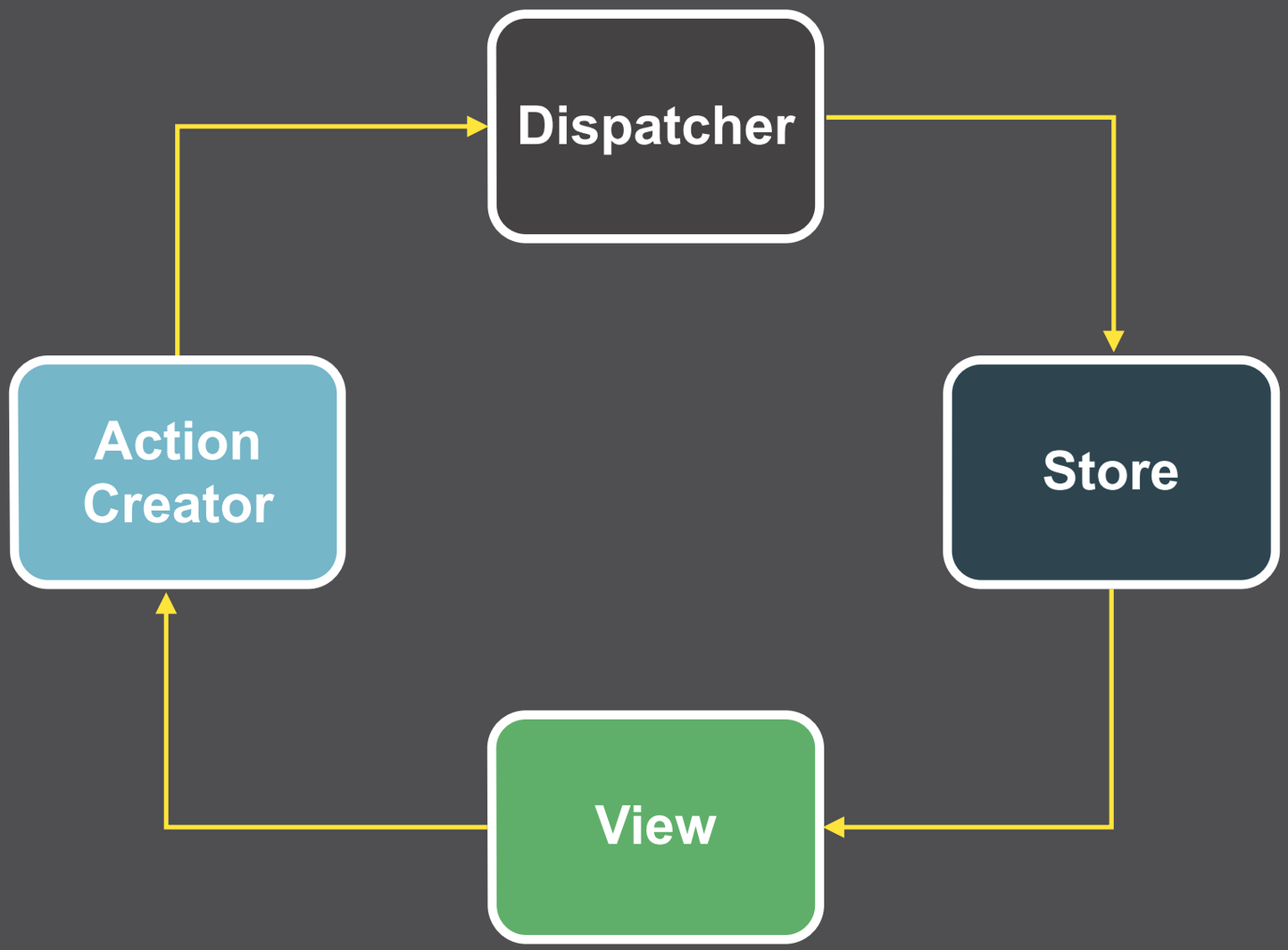
Hvordan?

Hvorfor?

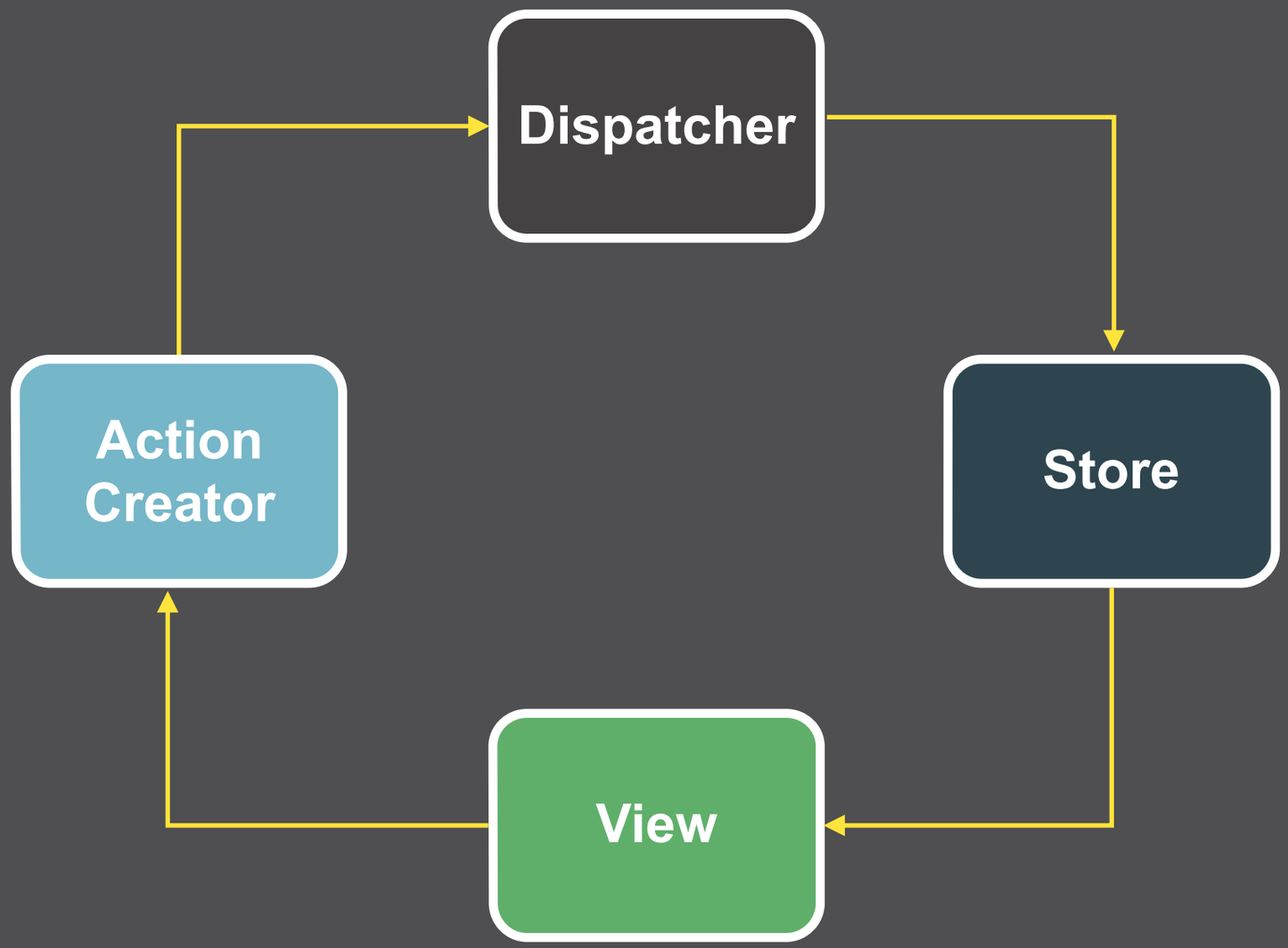
Hva?

Et design pattern for frontend

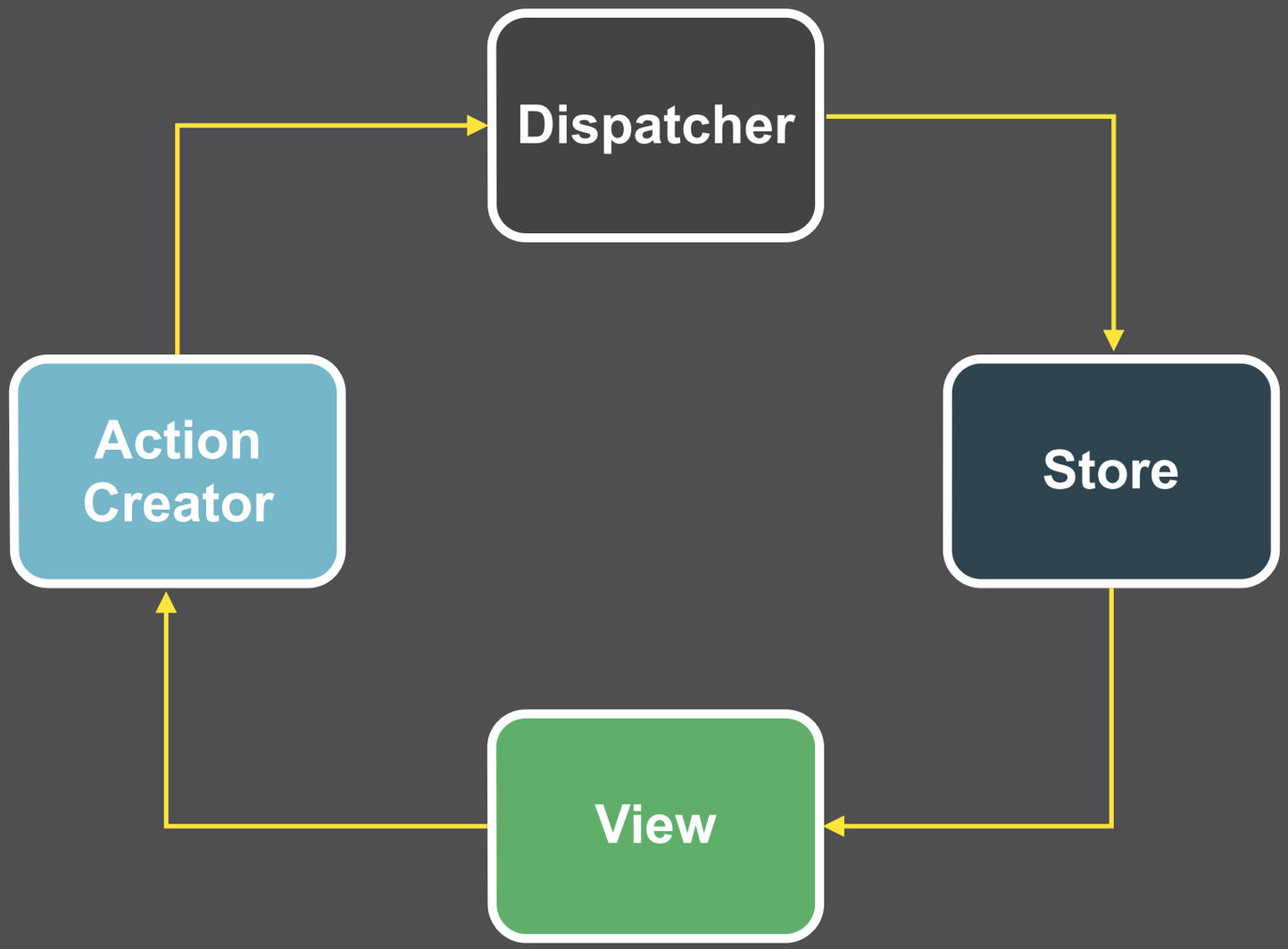
Flux



Dataflyten går kun én vei

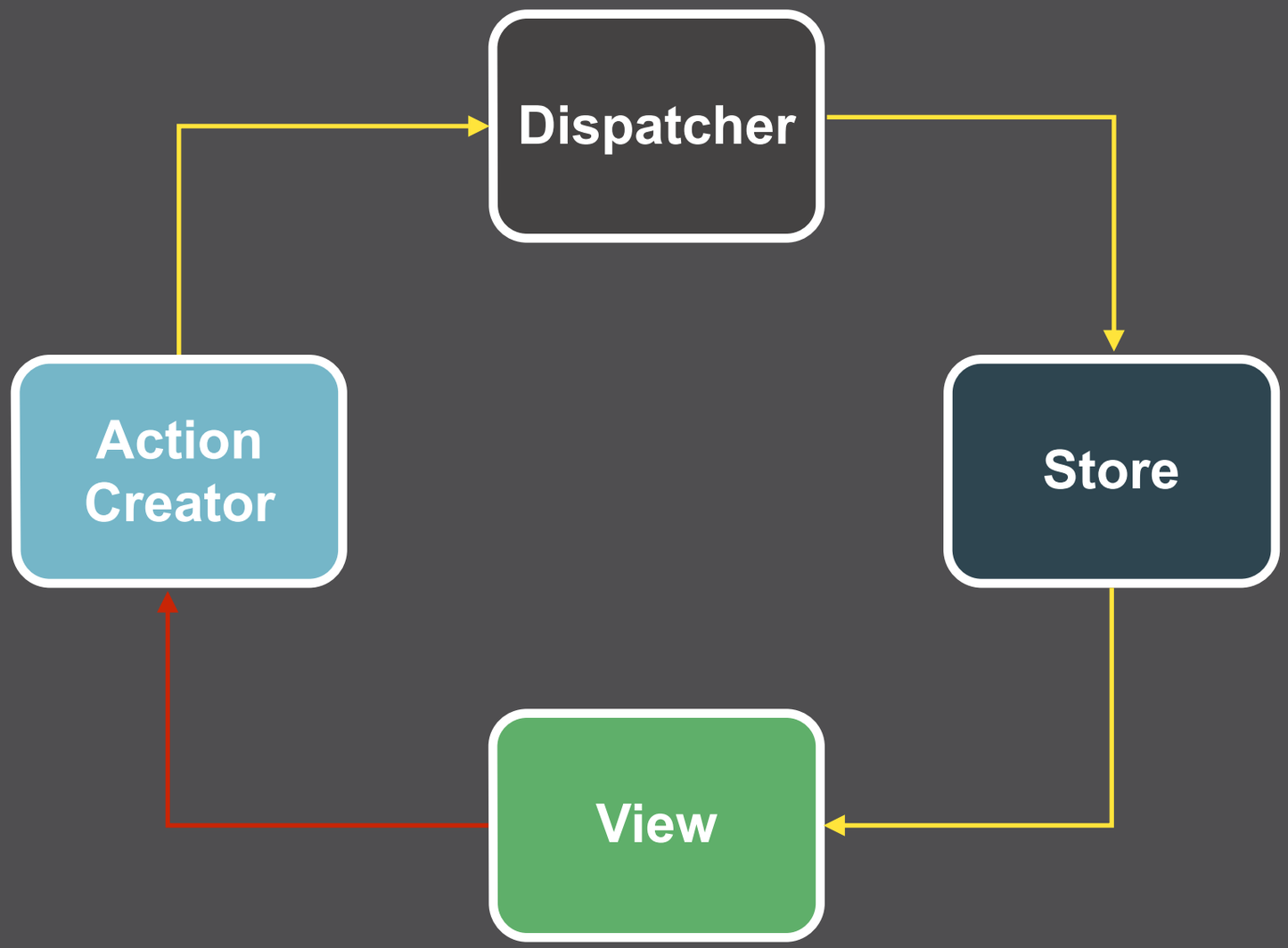


Mental modell

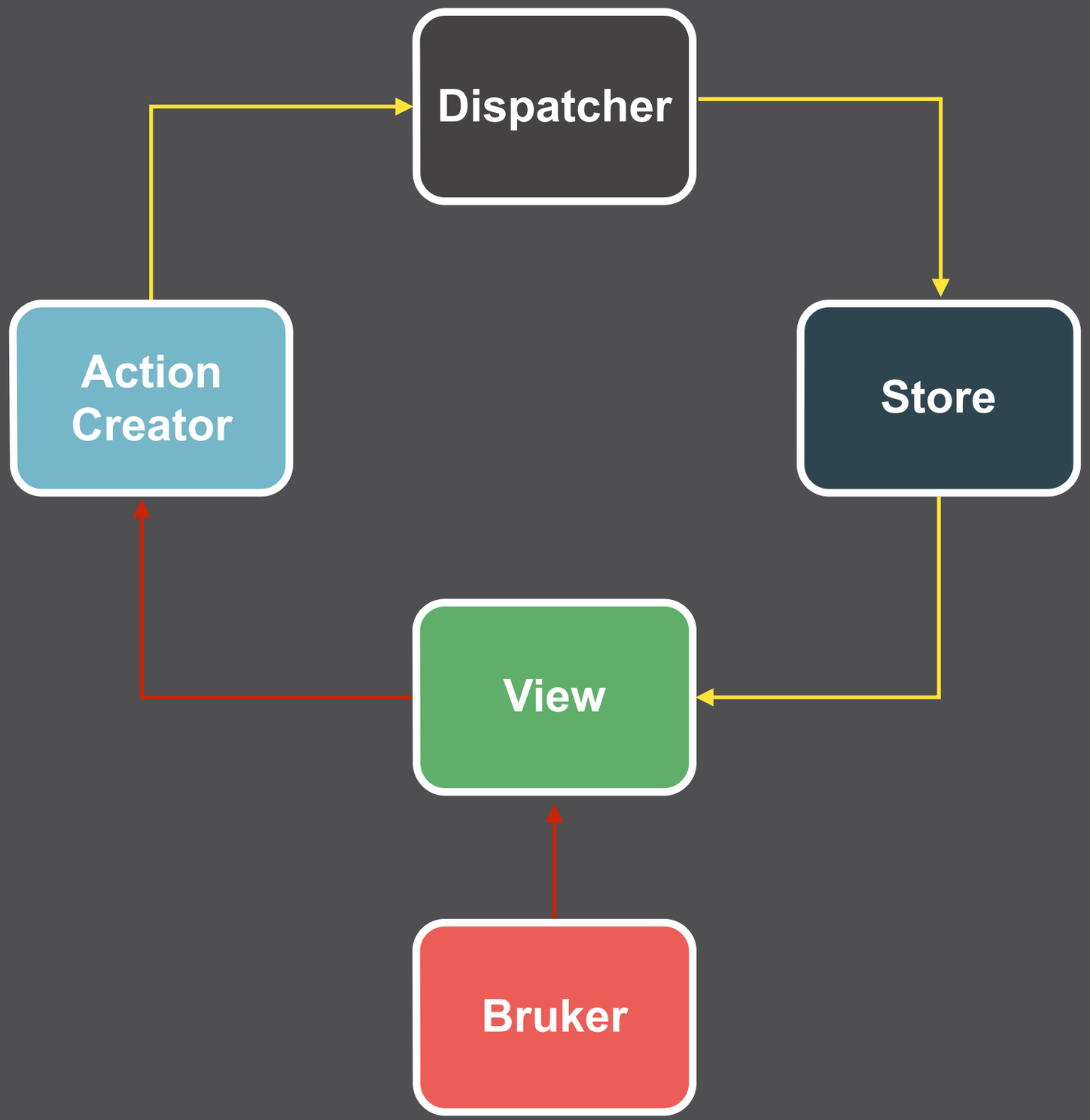


Hvordan?

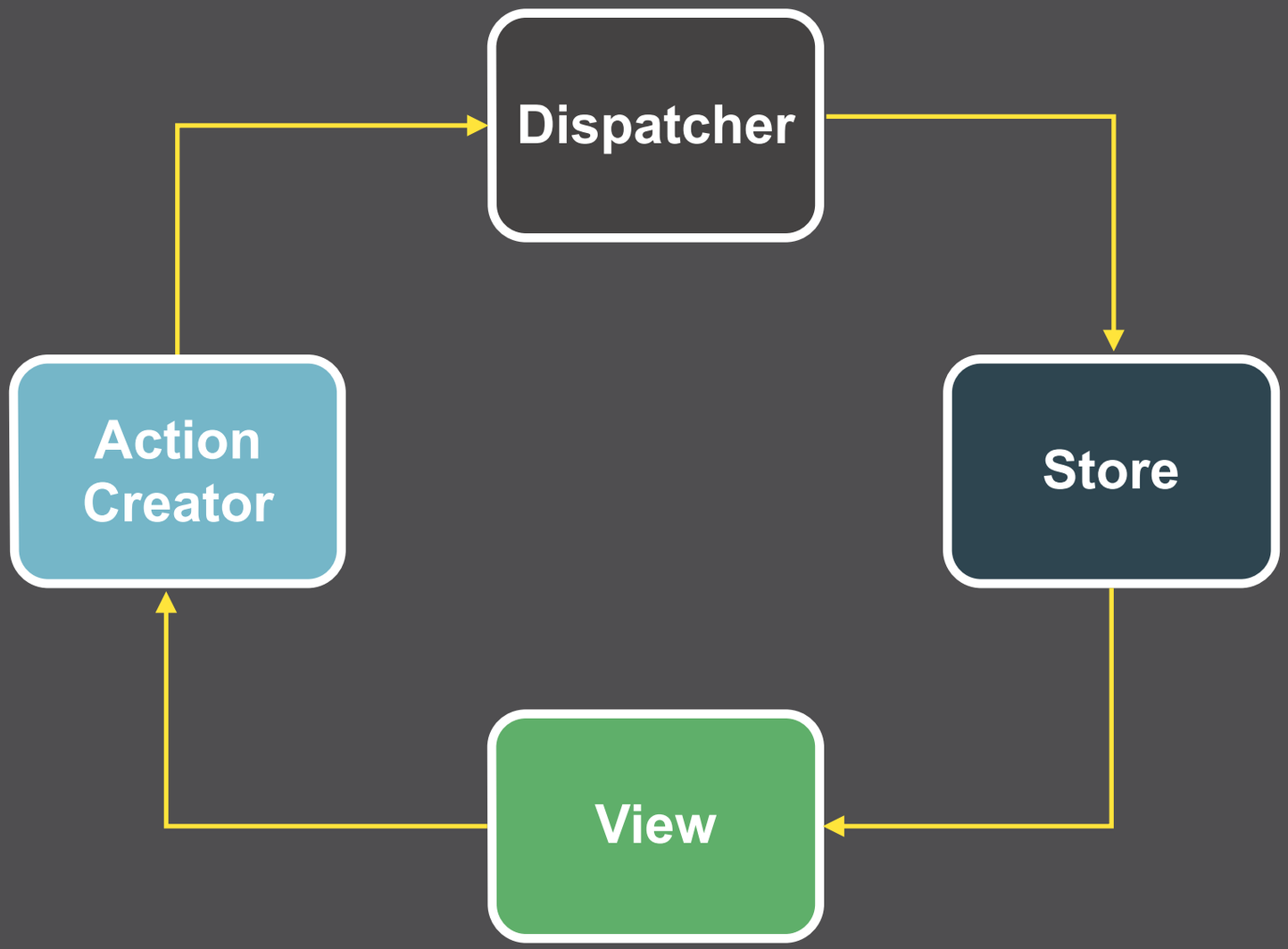
Hvor kommer data fra?



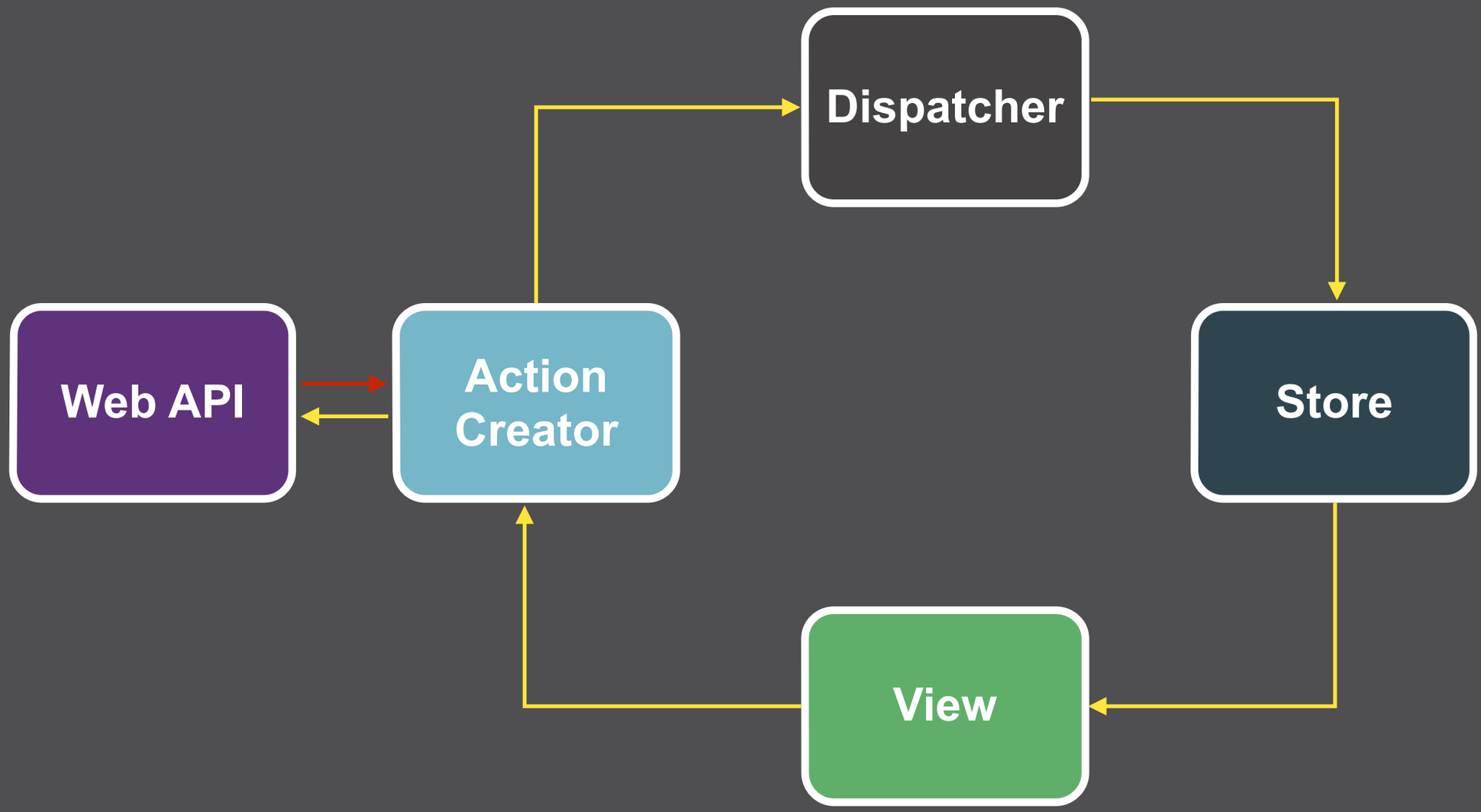
Hvor kommer data fra?



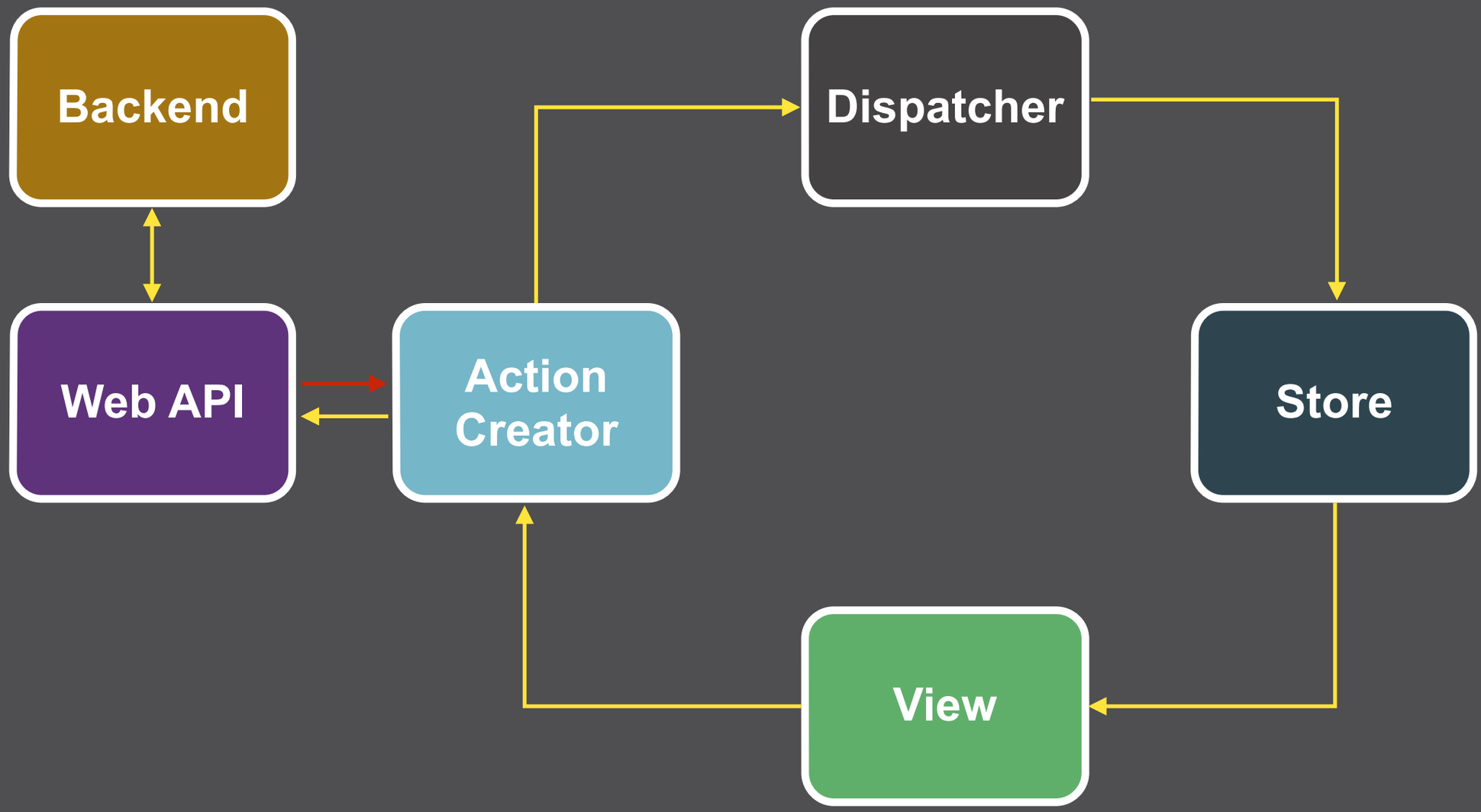
Data fra backend?



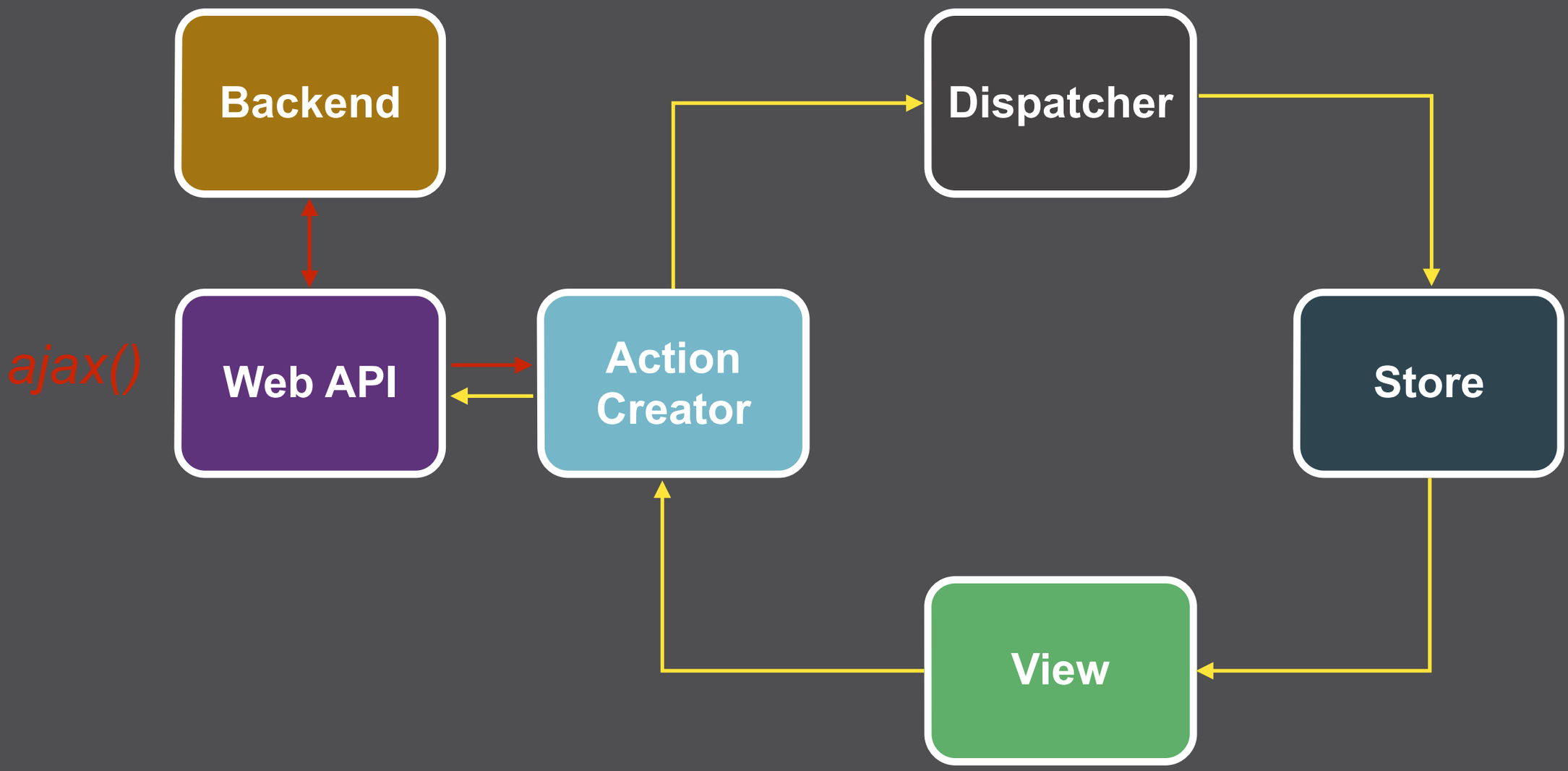
Data fra backend?



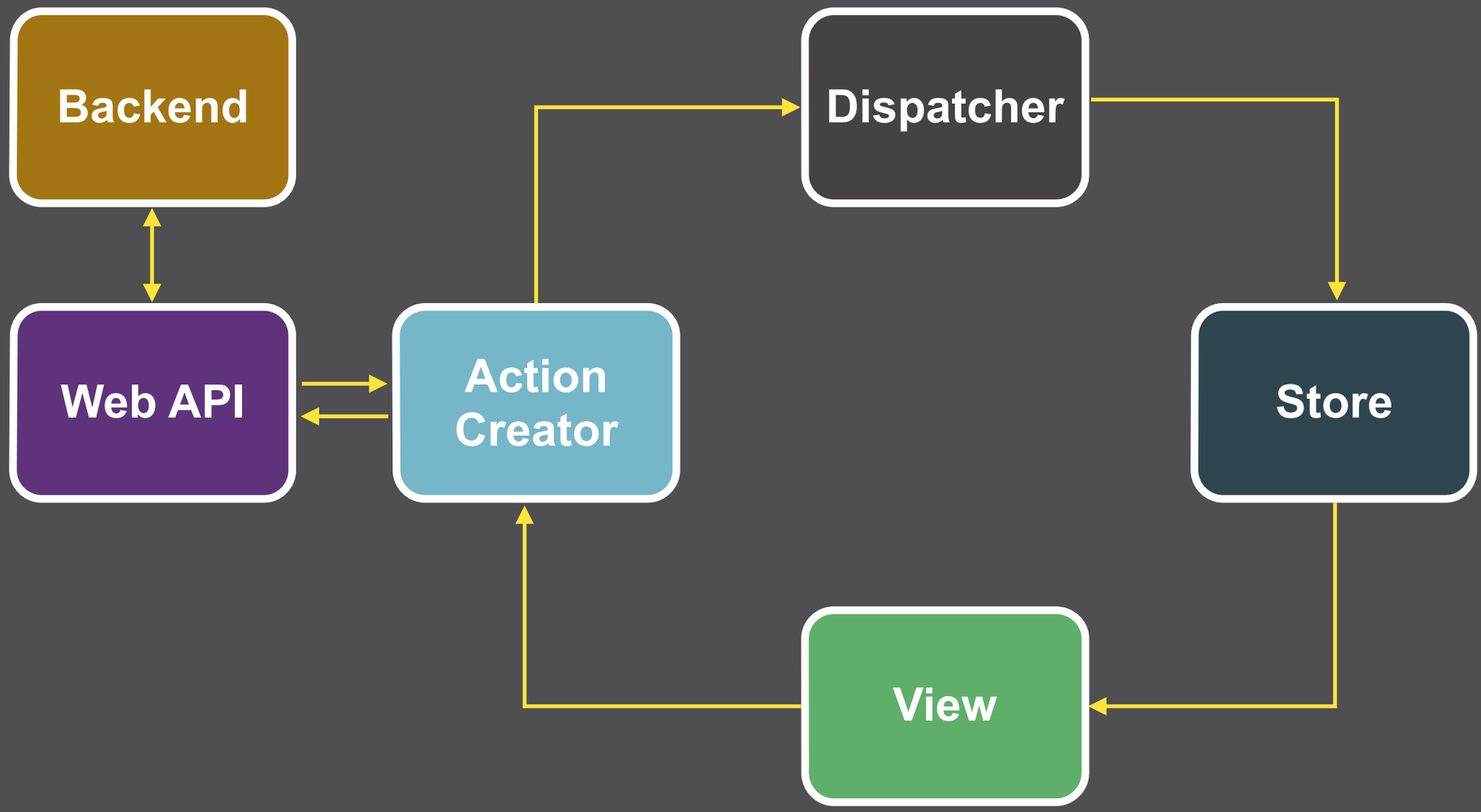
Data fra backend?



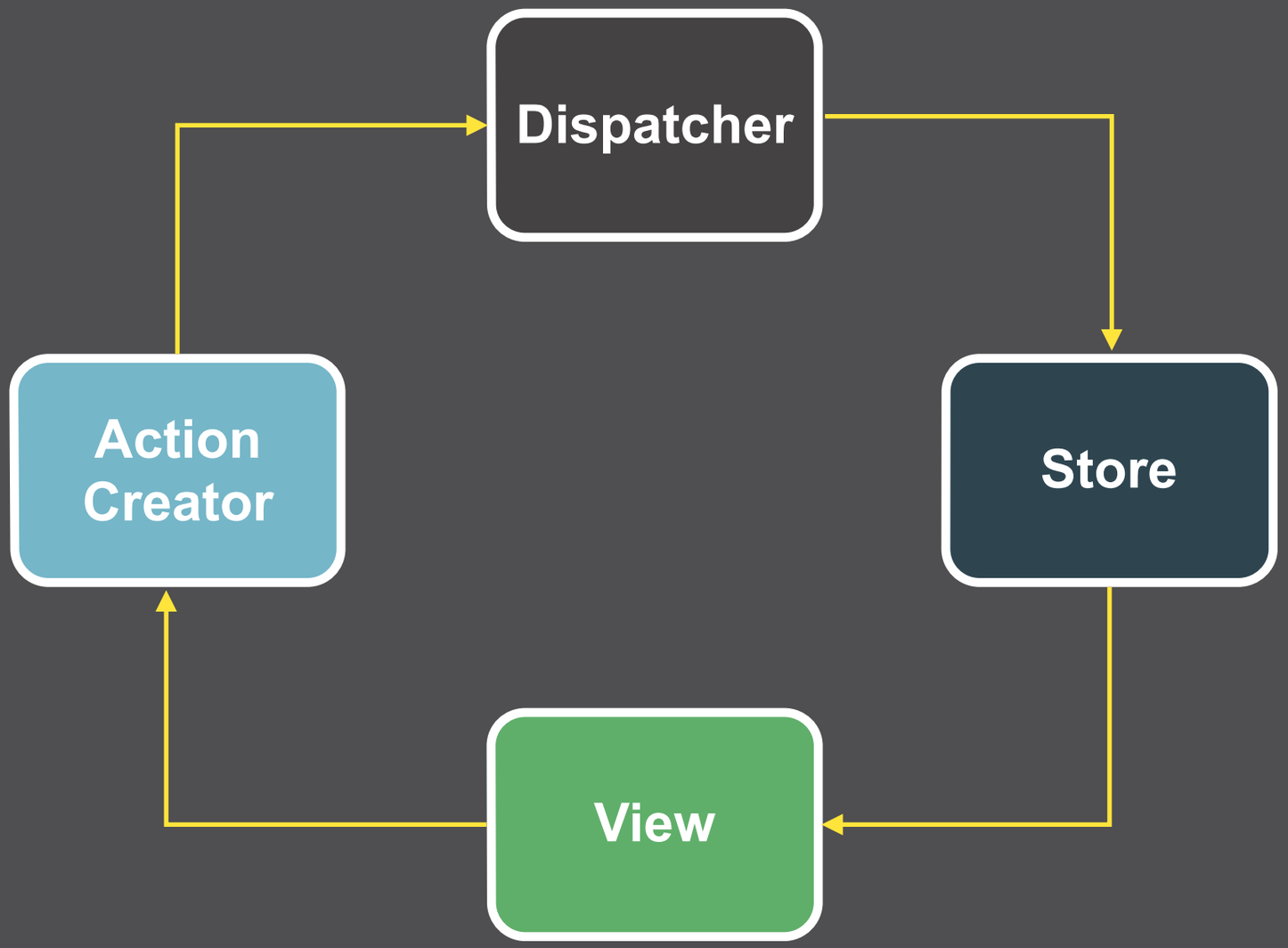
Data fra backend?



Hele arkitekturen

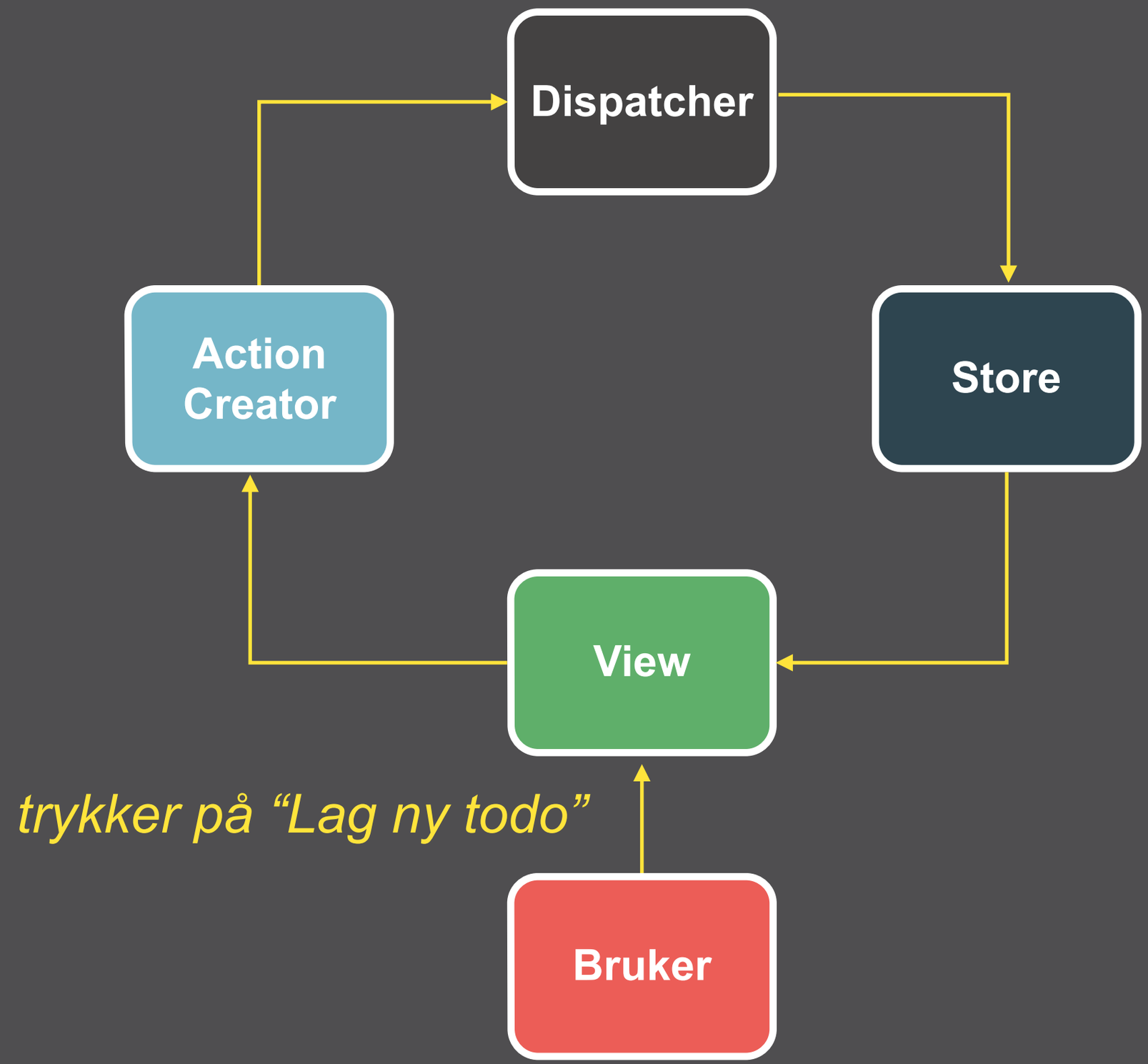


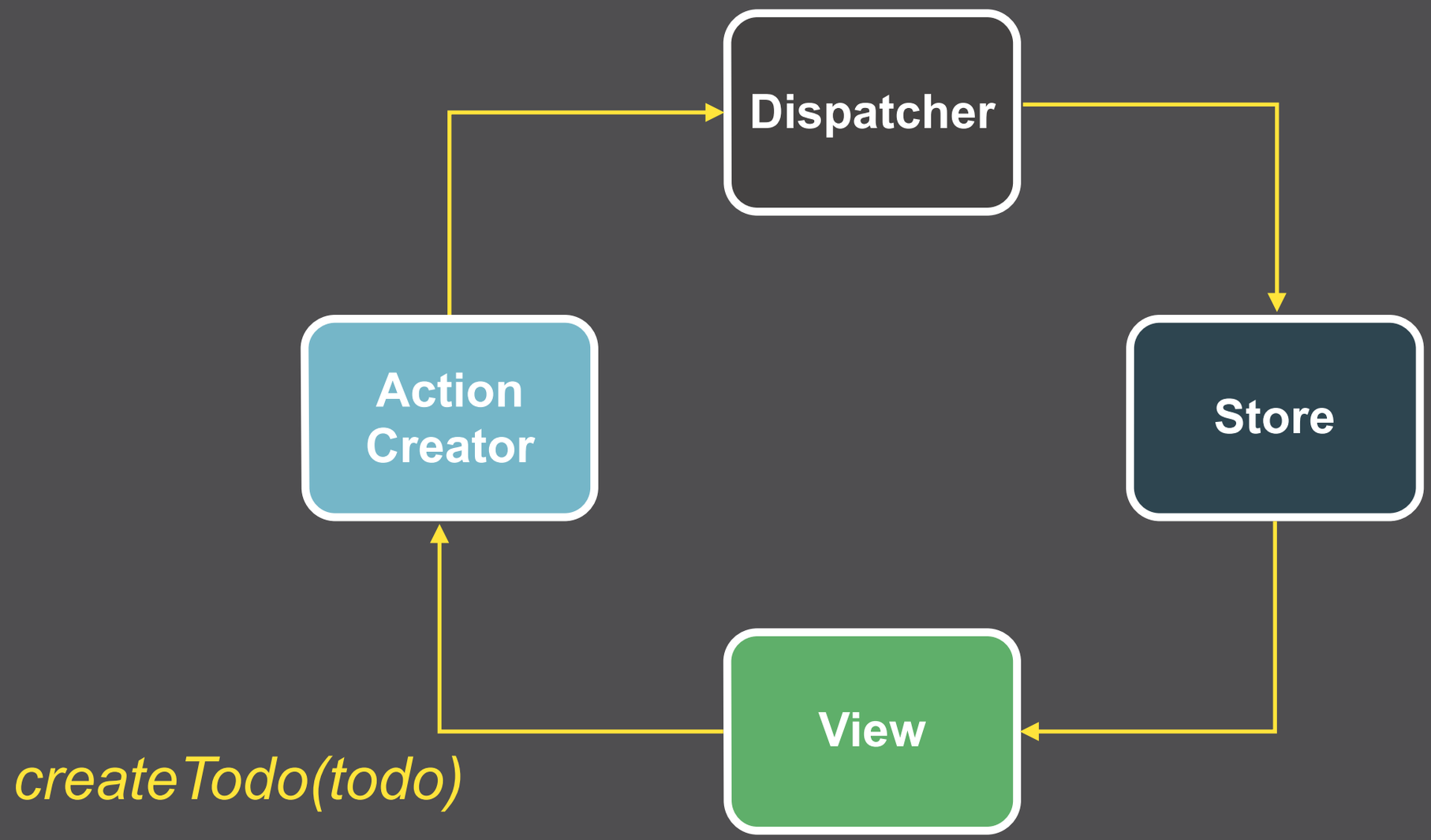
Mental modell

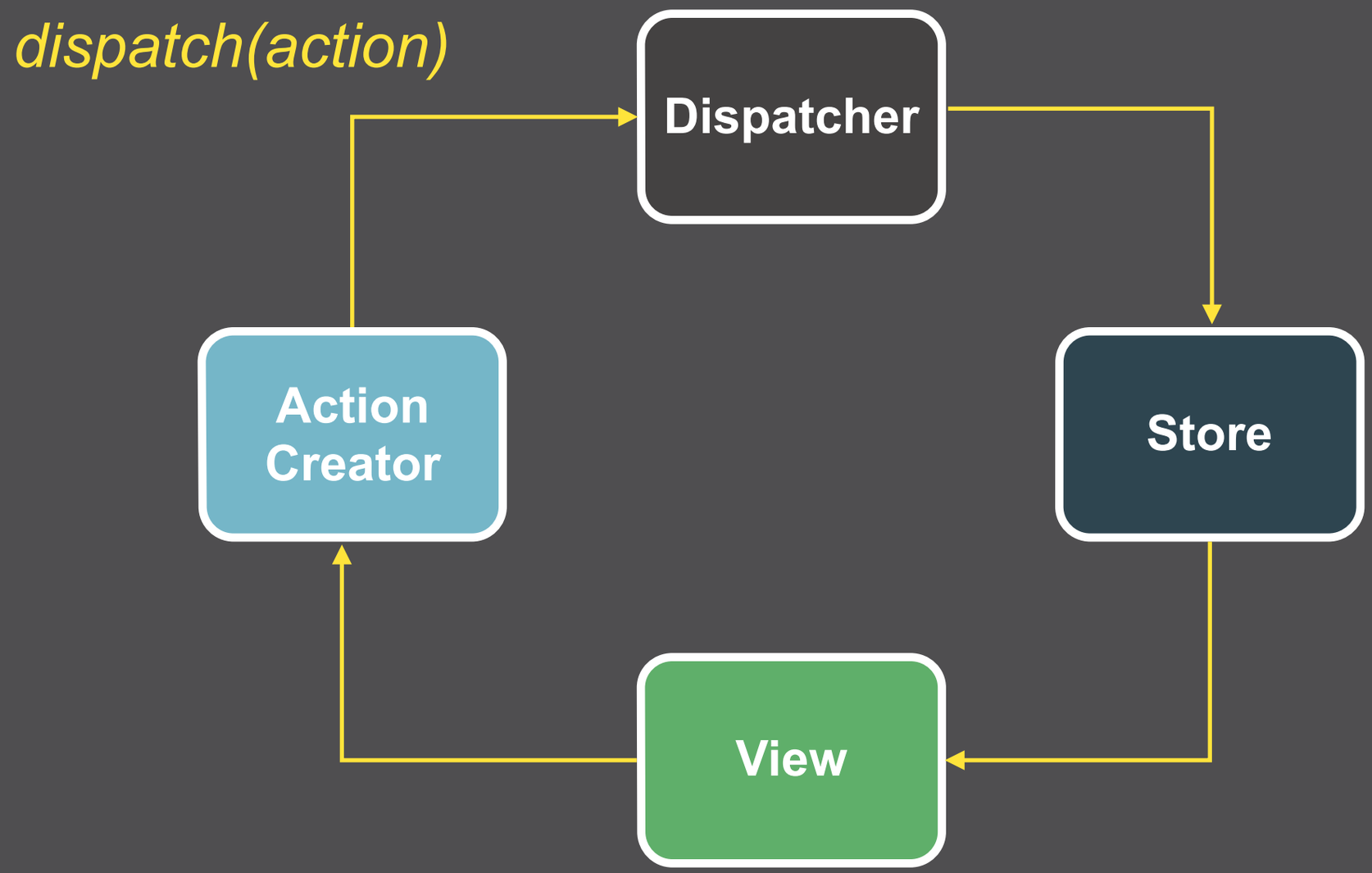


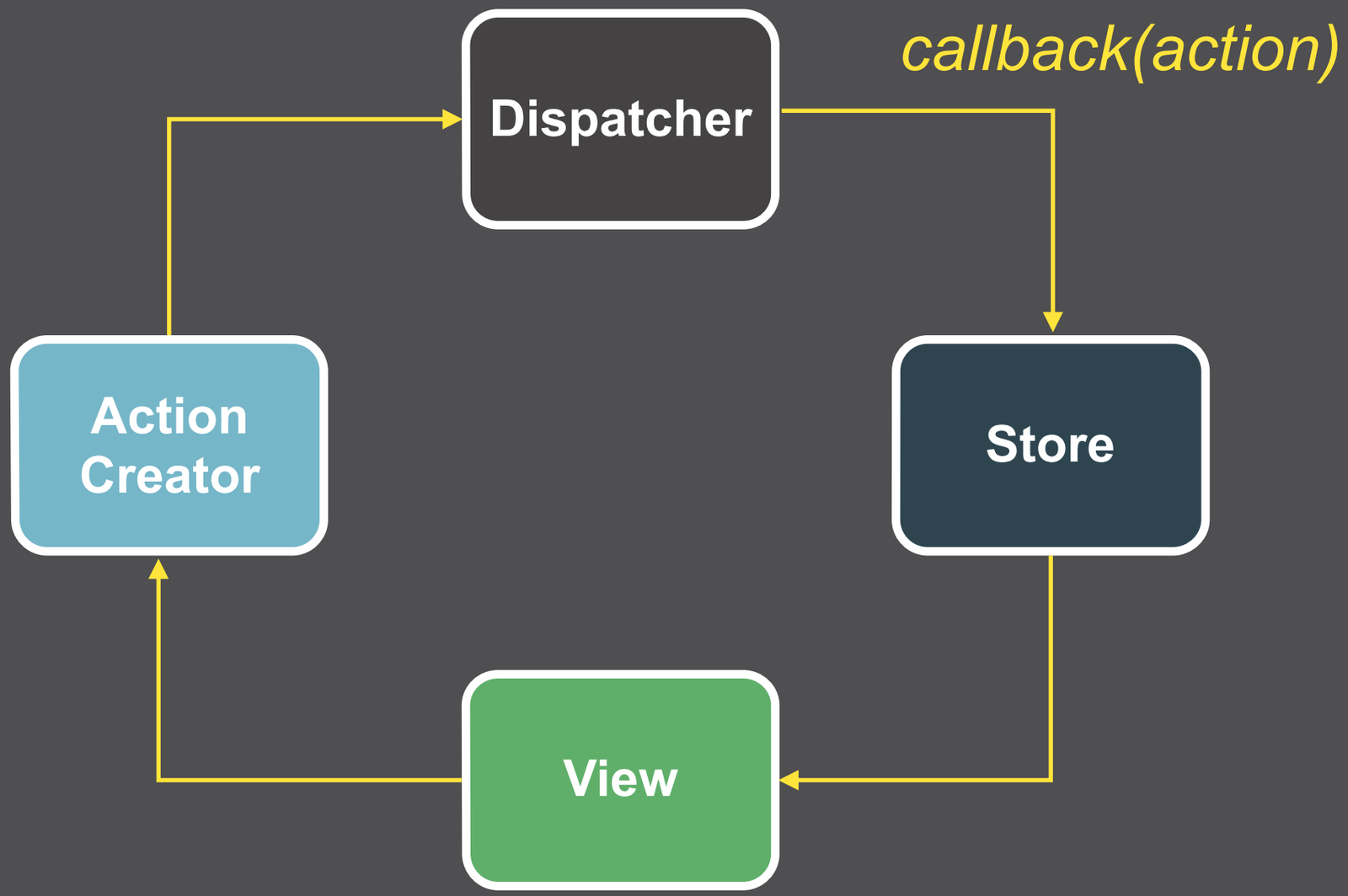
Eksempelscenario

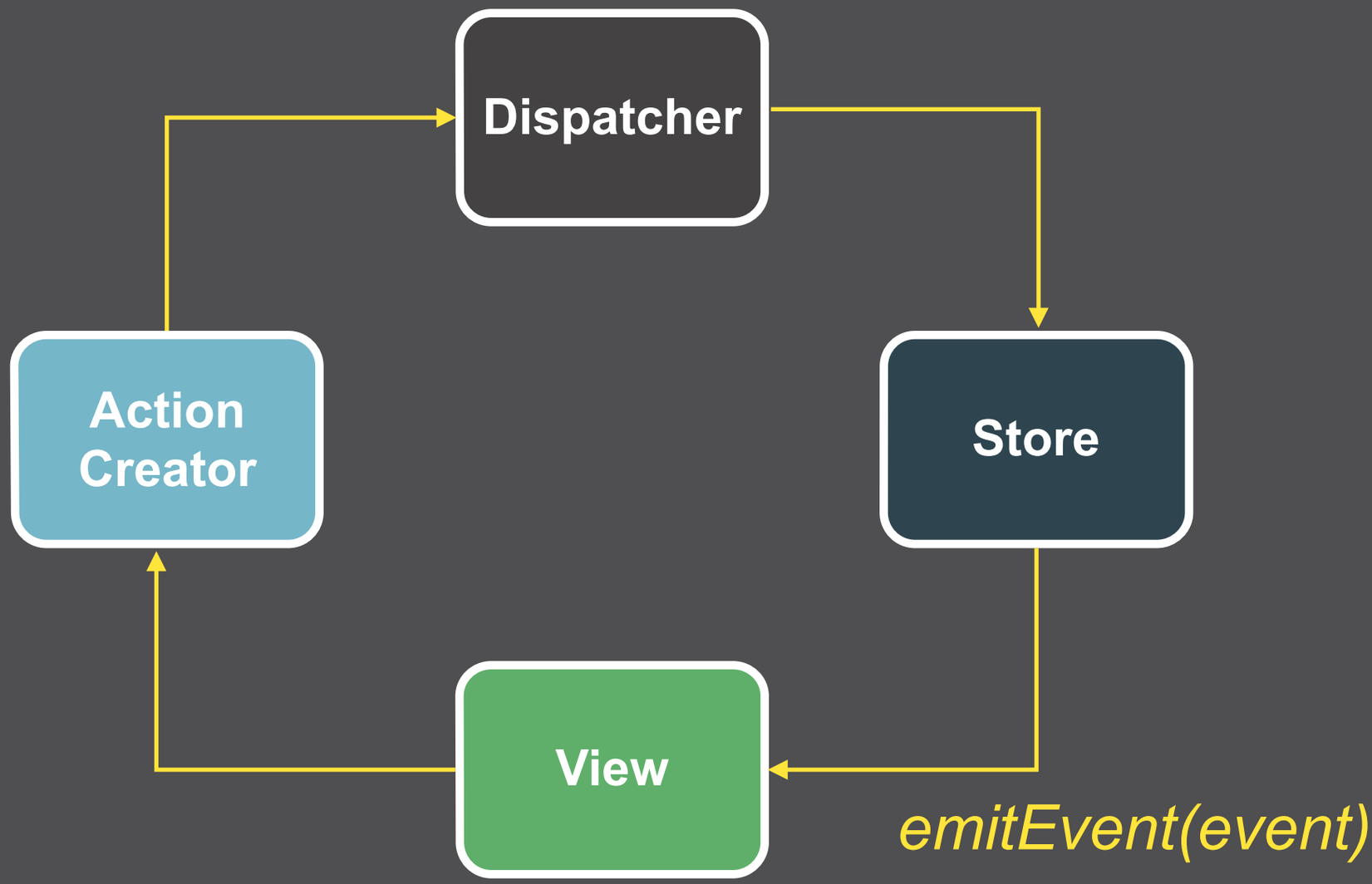
En Todo-App!

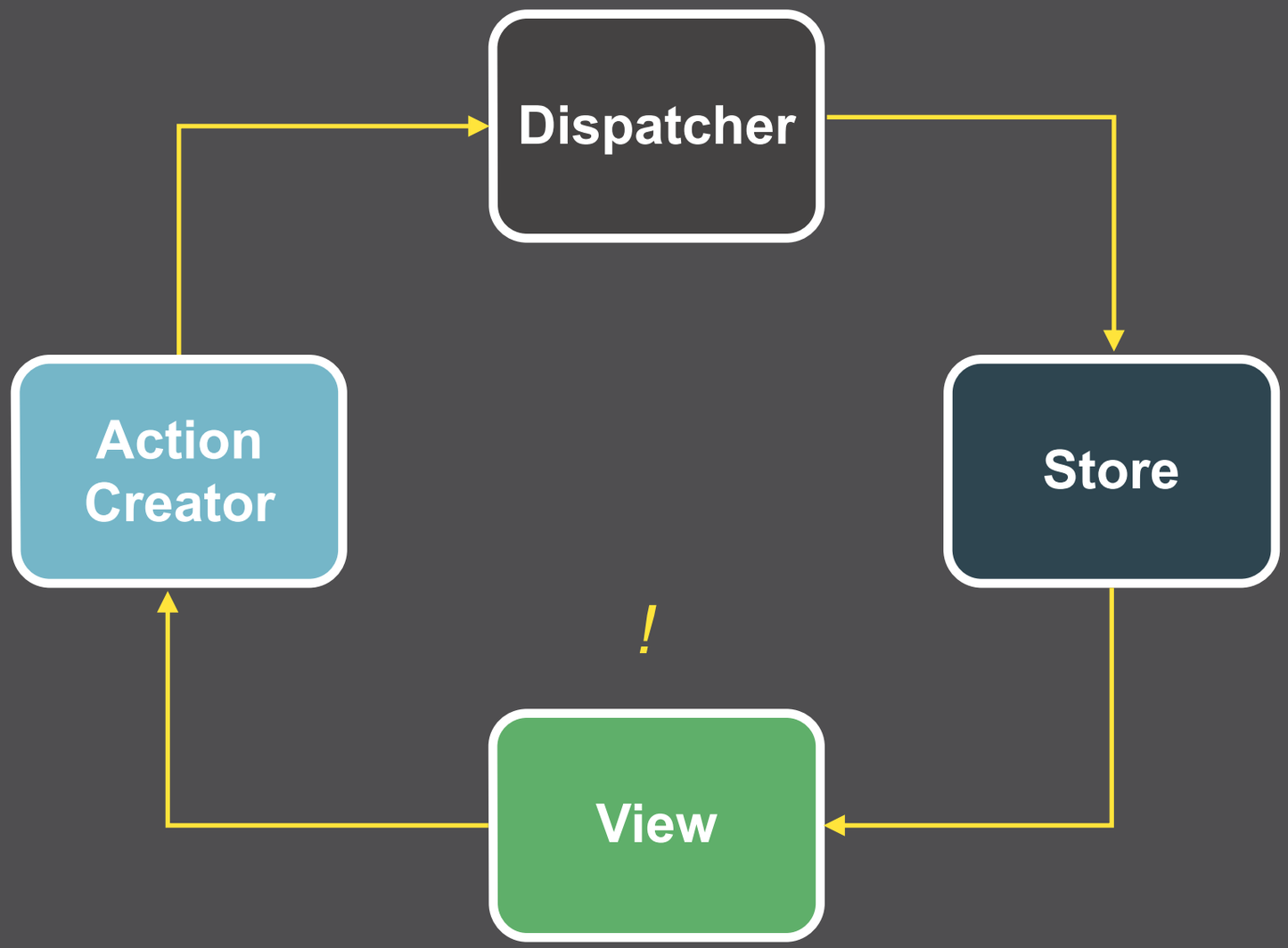

Dispatcher
Action Creator
Store
View
Bruker
trykker på “Lag ny todo”

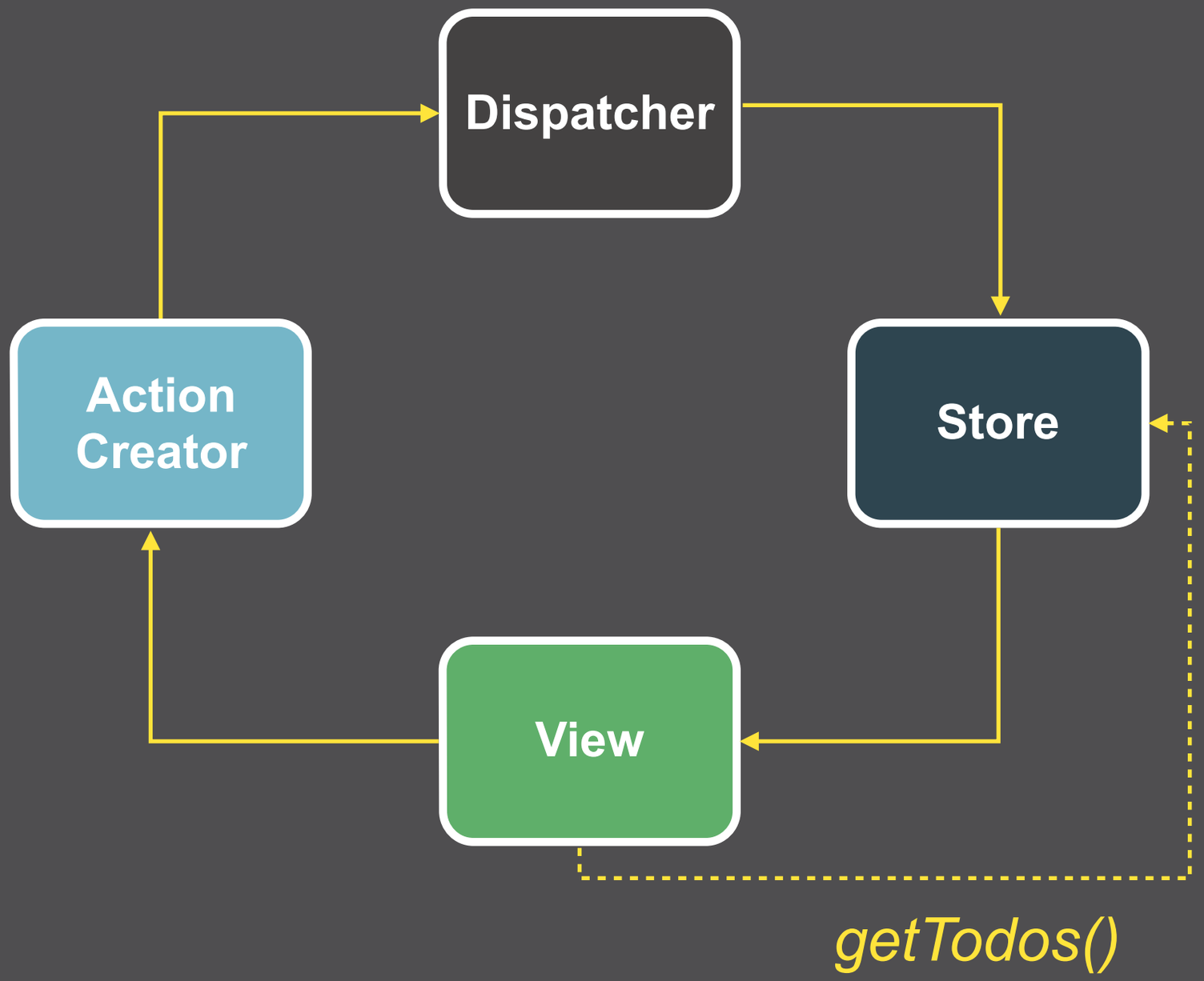

Dispatcher
Action Creator
Store
View
createTodo(todo)

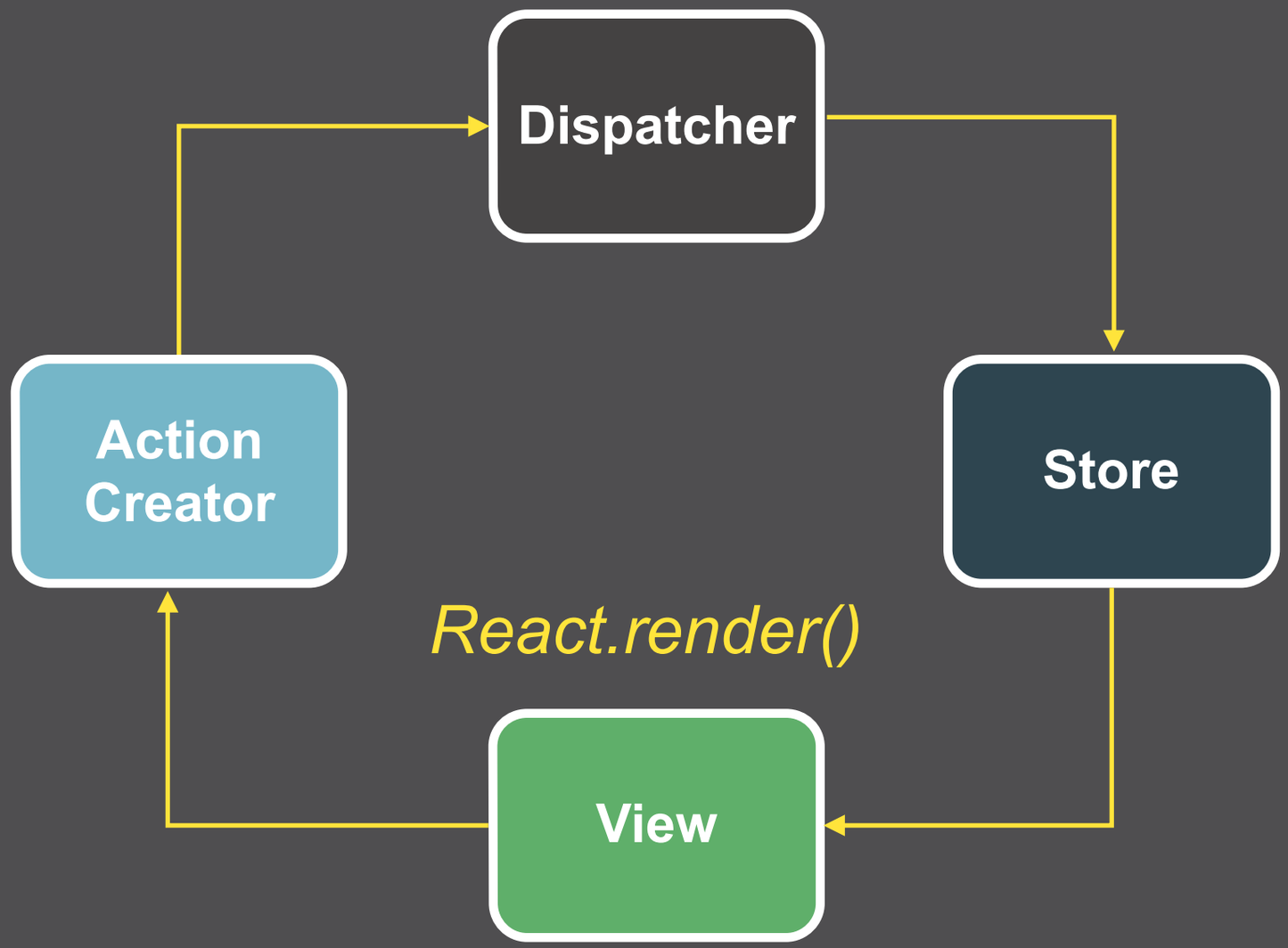

dispatch(action)
Dispatcher
Action Creator
Store
View

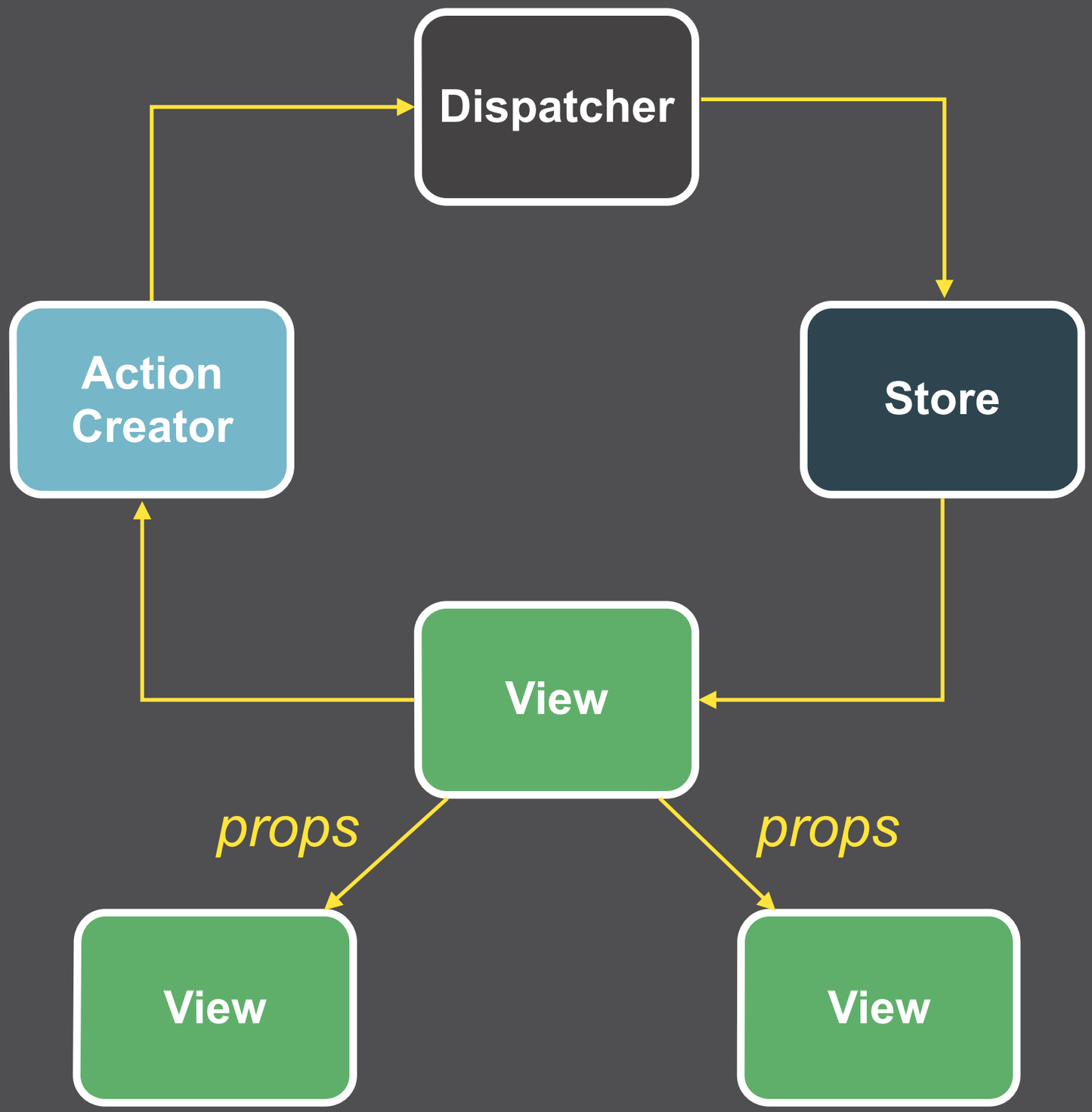

Dispatcher
callback(action)
Action Creator
Store
View

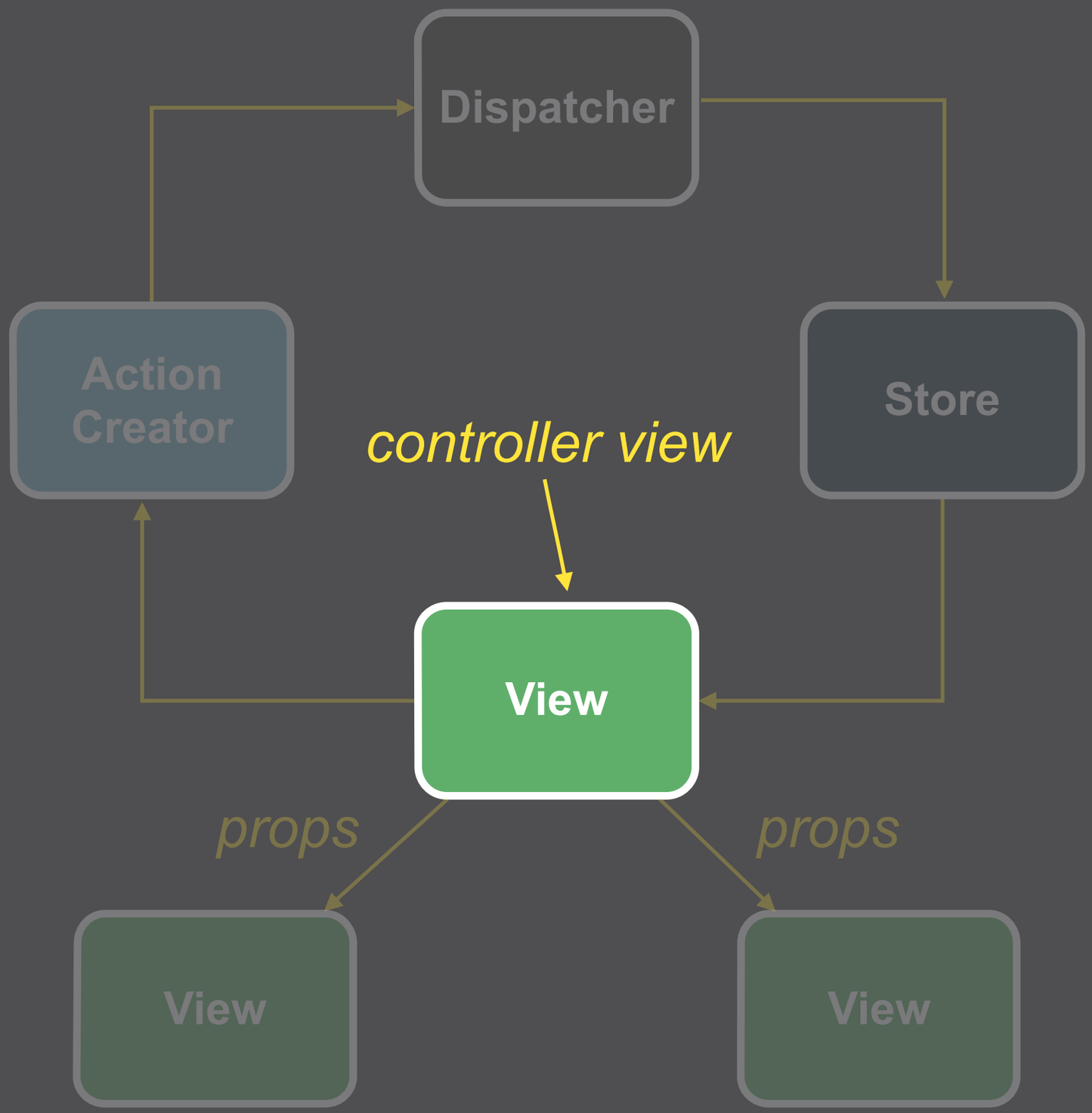

Dispatcher
Action Creator
Store
View
emitEvent(event)


Dispatcher
Action Creator
Store
!
View

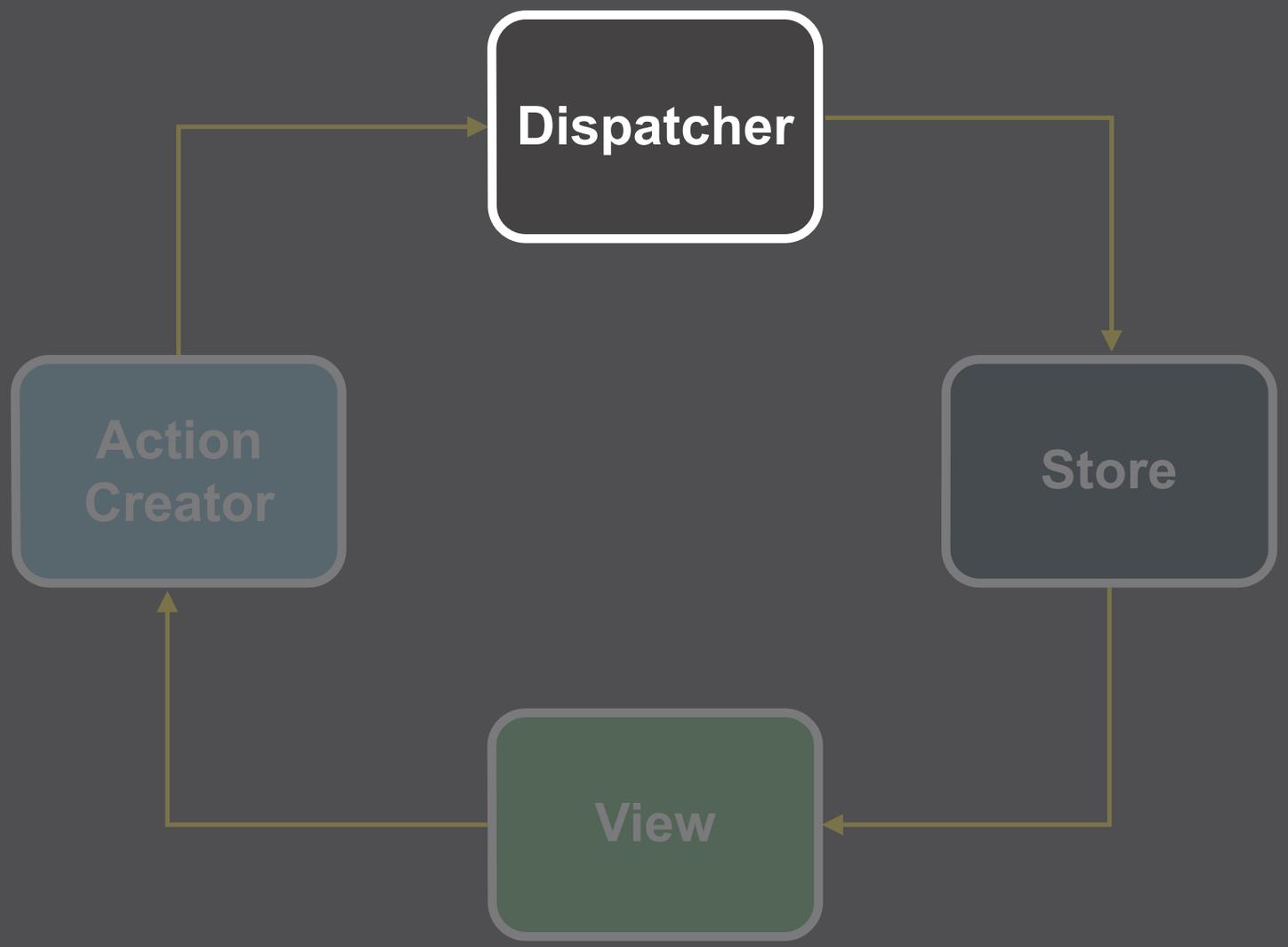

Dispatcher
Action Creator
Store
View
getTodos()

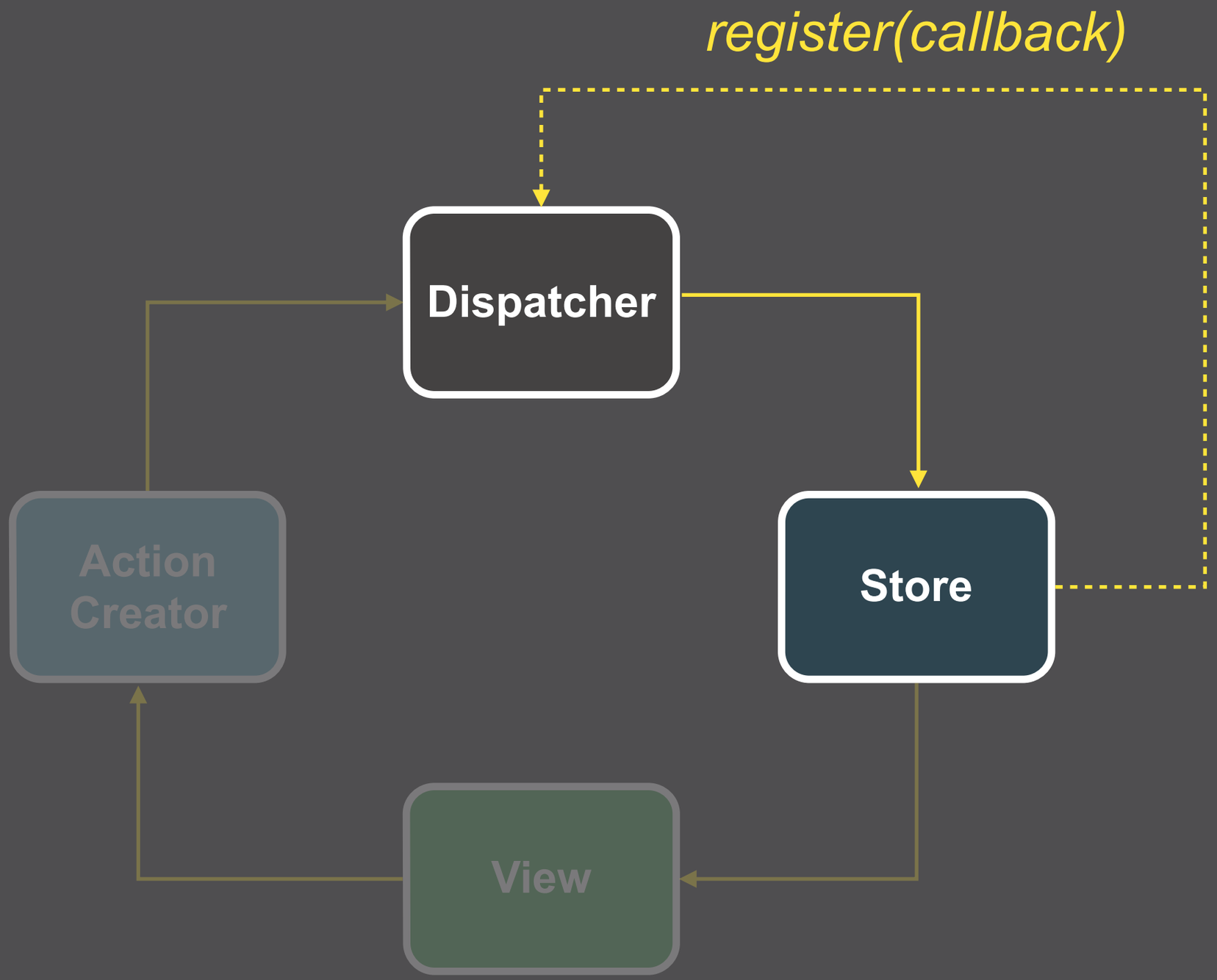

Dispatcher
Action Creator
Store
React.render()
View


Dispatcher
Action Creator
Store
View
props
props
View
View

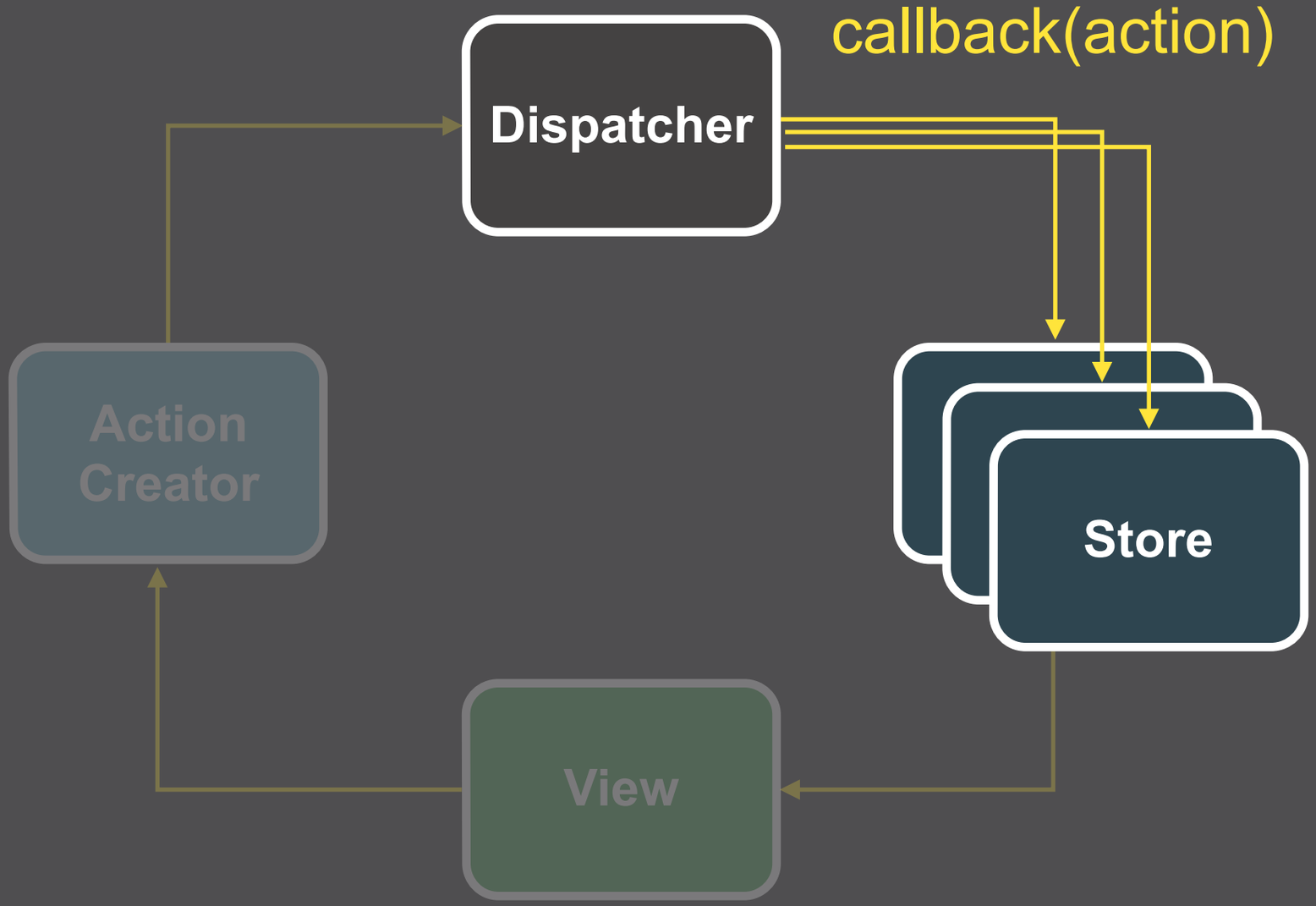

Dispatcher
Action Creator
Store
controller view
View
props
props
View
View

Sentrale komponenter

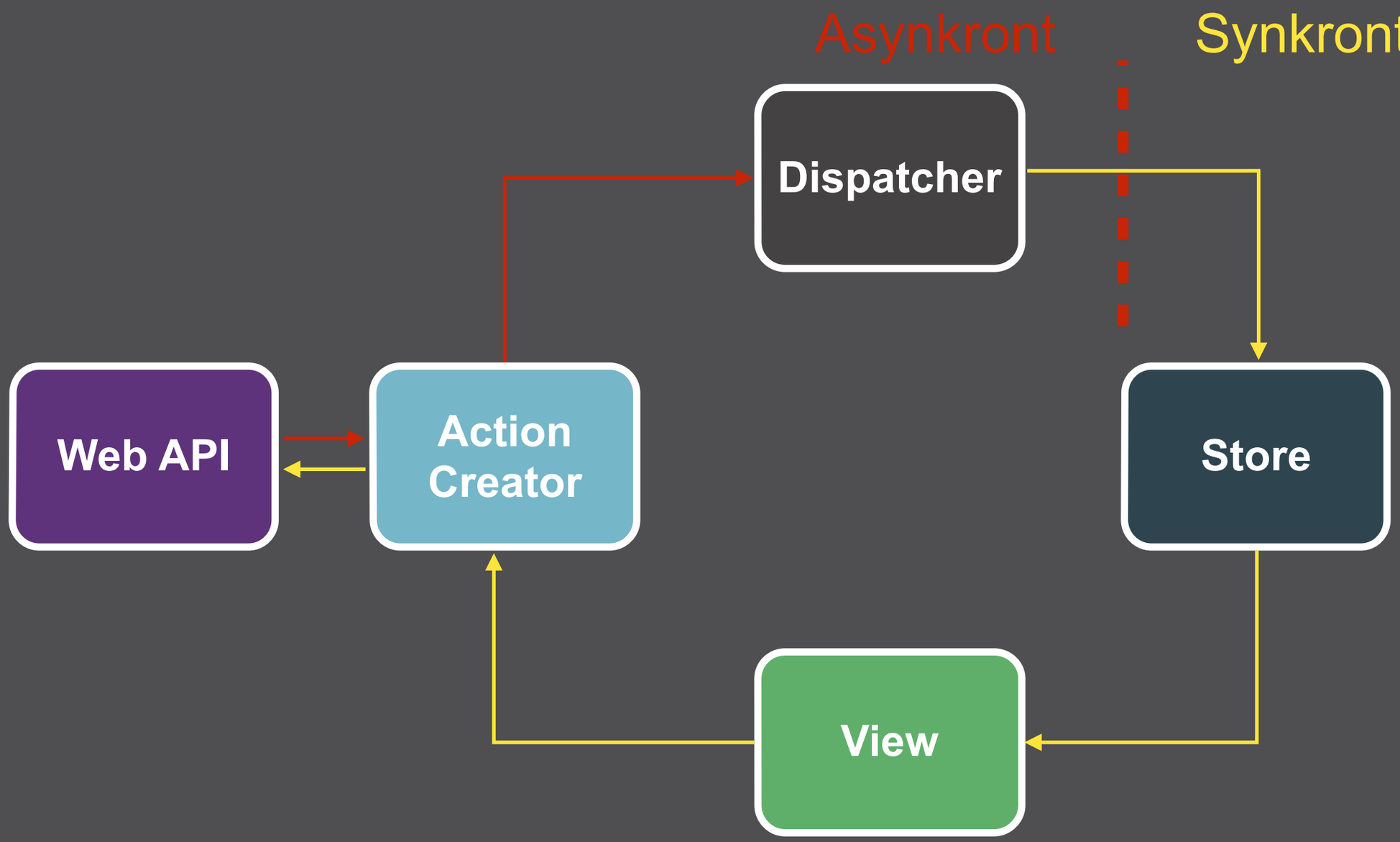

Dispatcher
Action Creator
Store
View


register(callback)
Dispatcher
Action Creator
Store
View

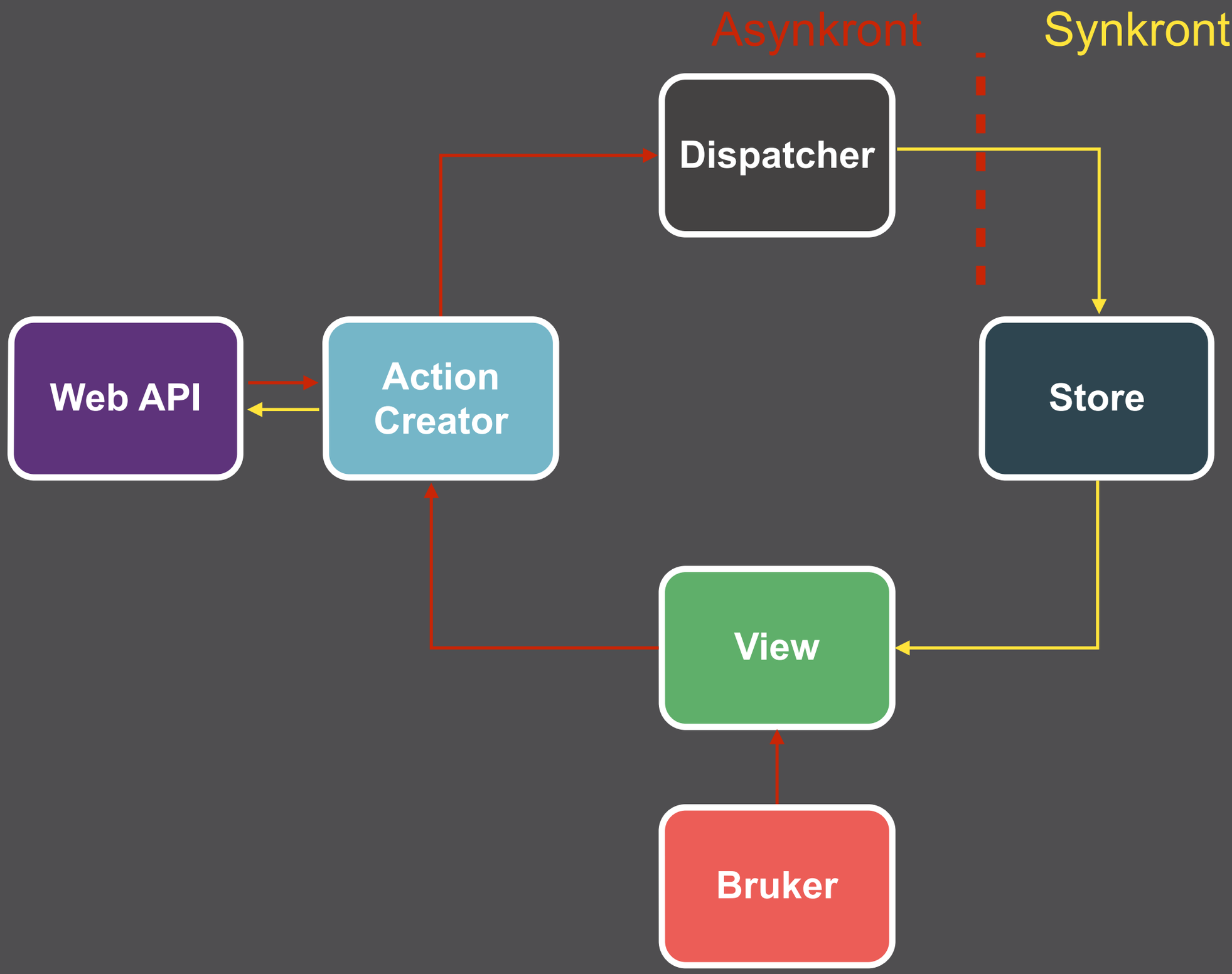
Synkron broadcasting



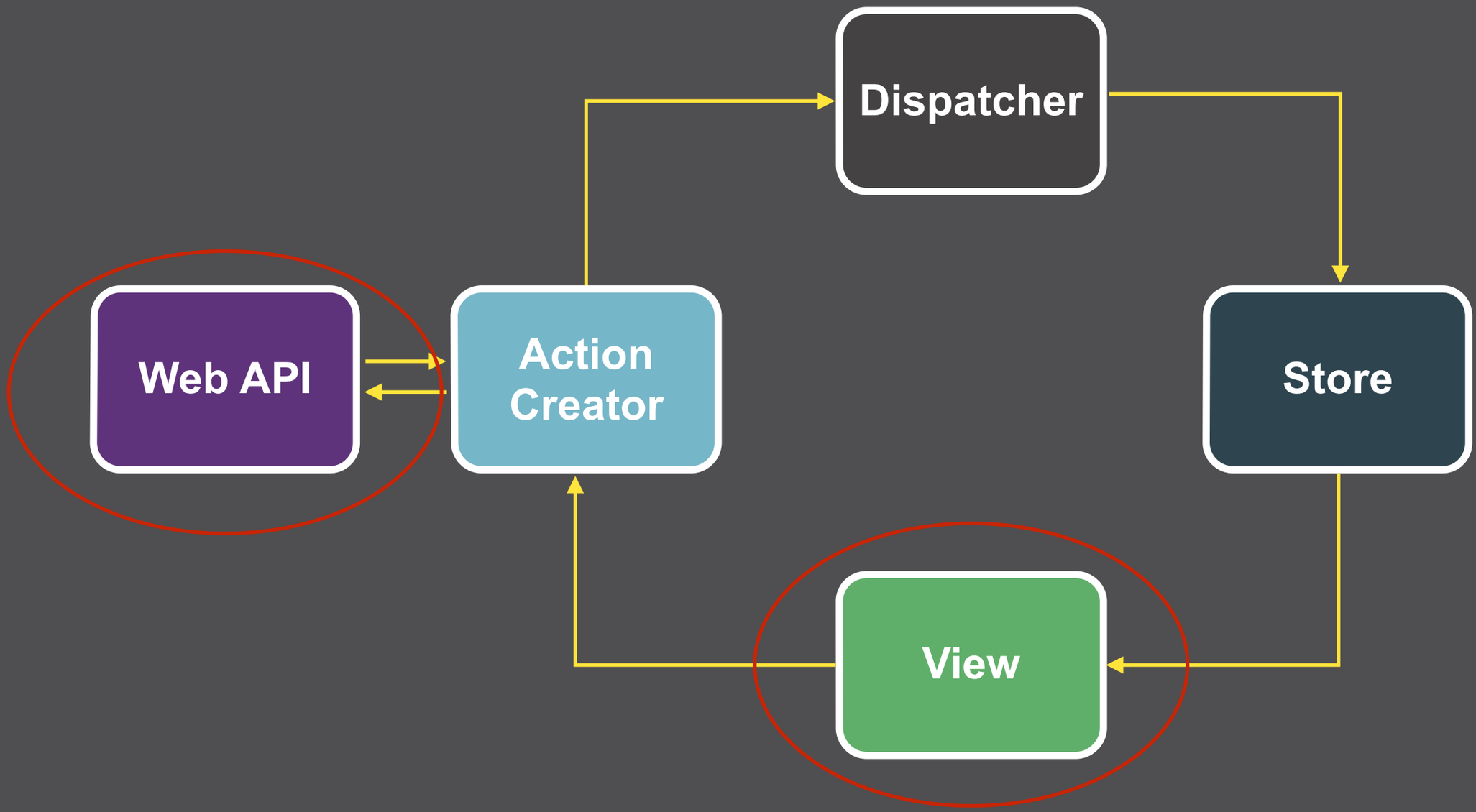
Async “barriere”



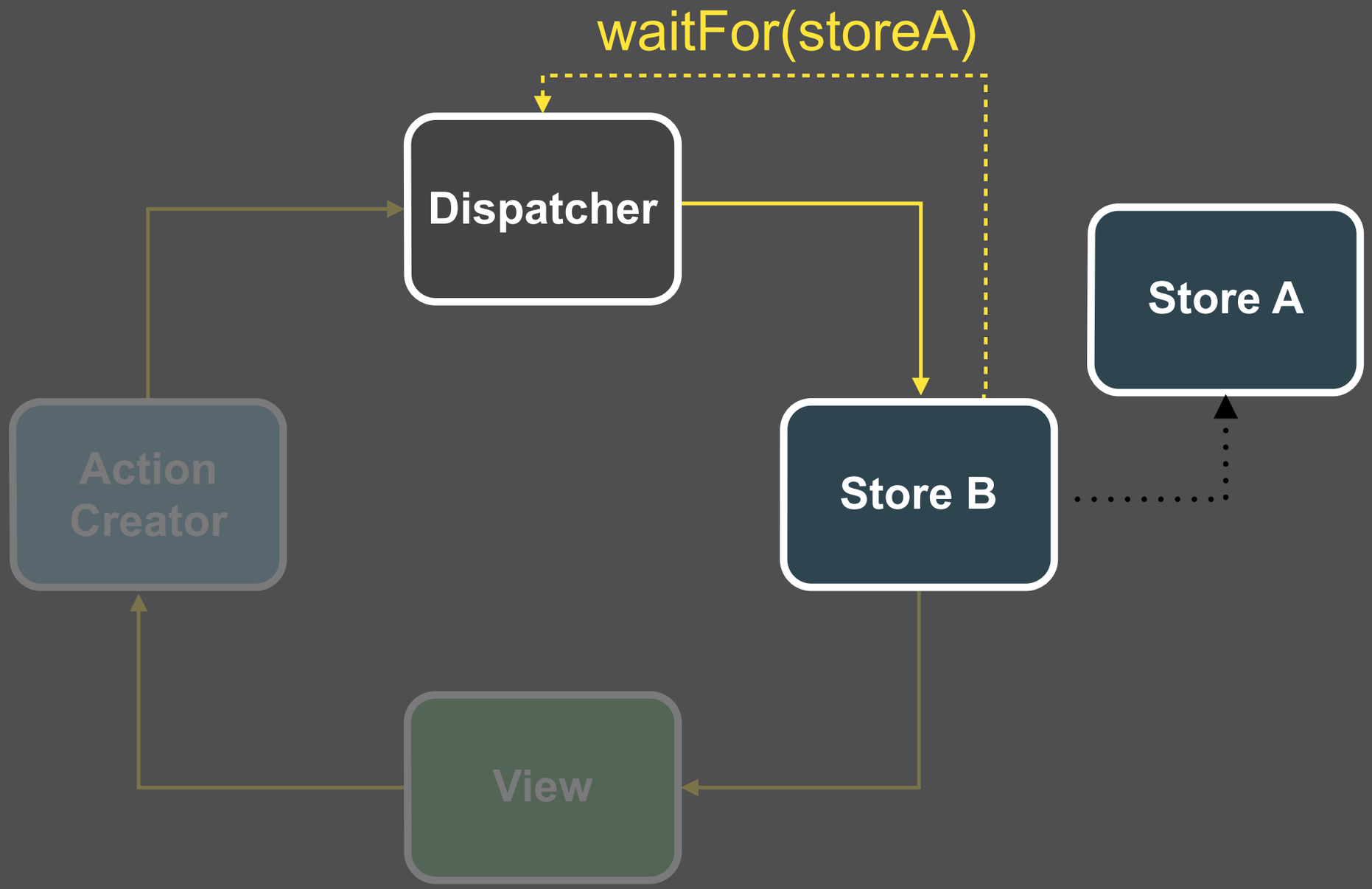
Async “barriere”



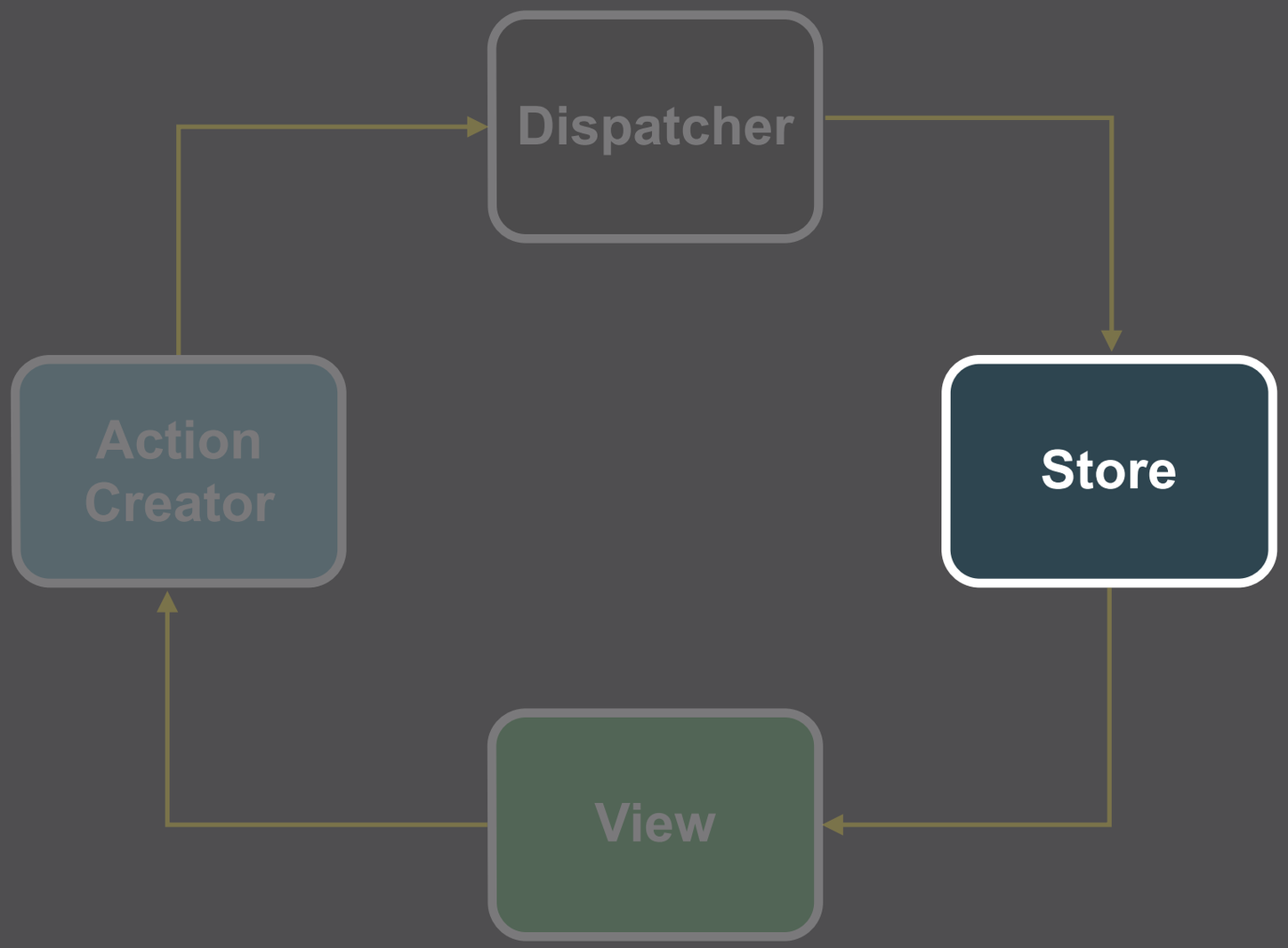
Isolering av async



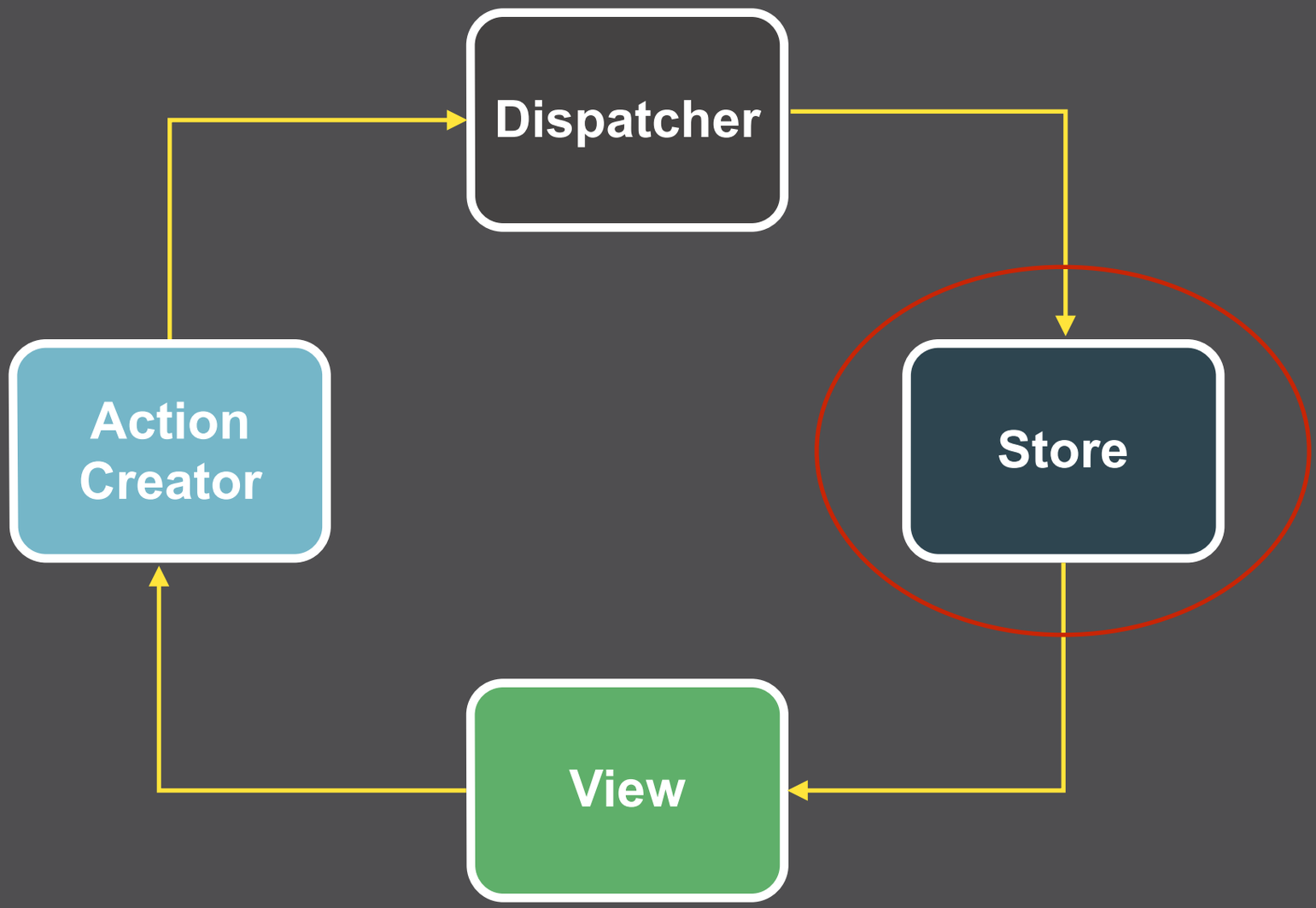
Avhengigheter



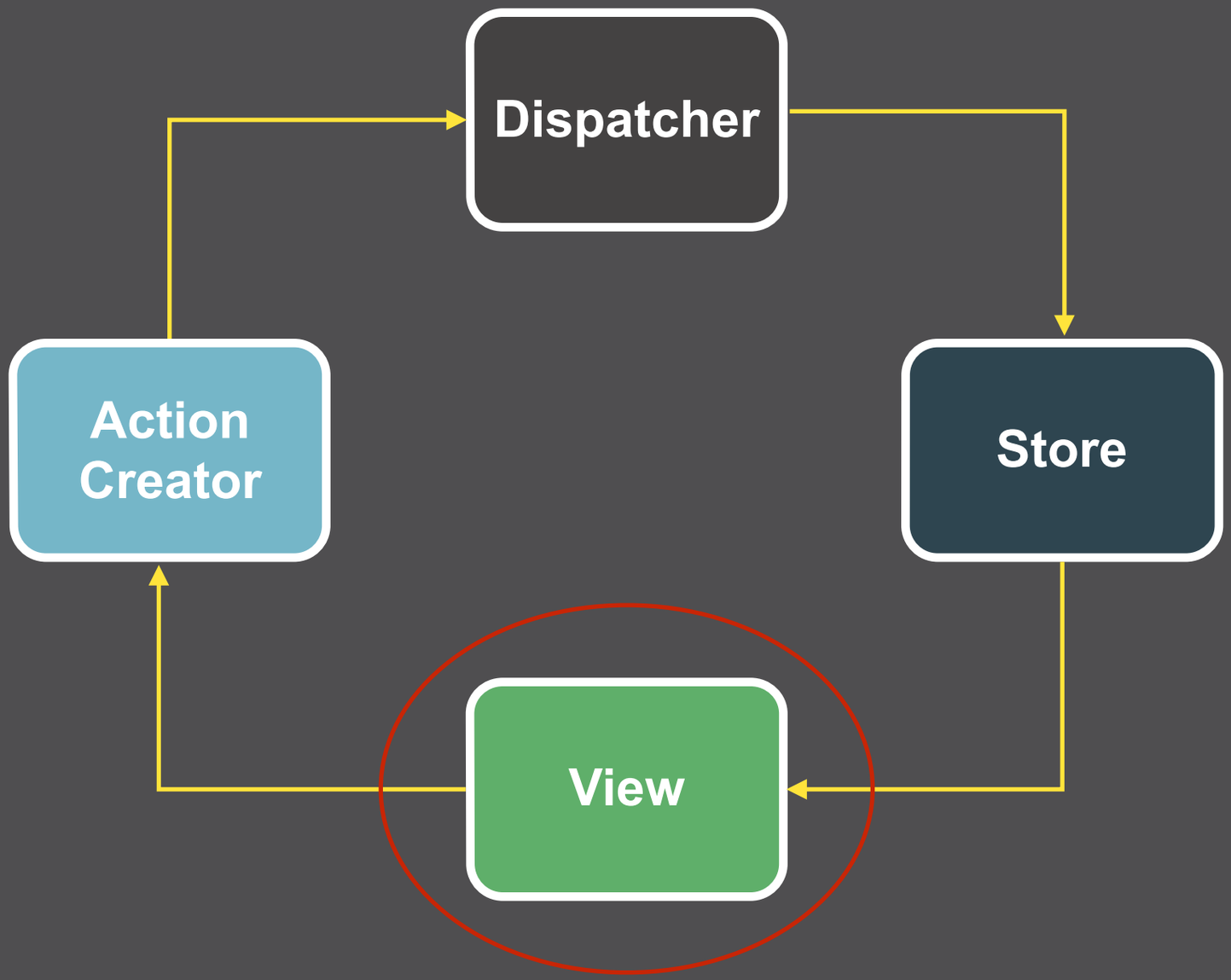
- Register over callbacks i stores
- Barriere mellom asynkront/synkront
- Kan håndtere avhengigheter mellom stores
- *dispatch(action), register(callback), waitFor(store)*


Dispatcher
Action Creator
Store
View

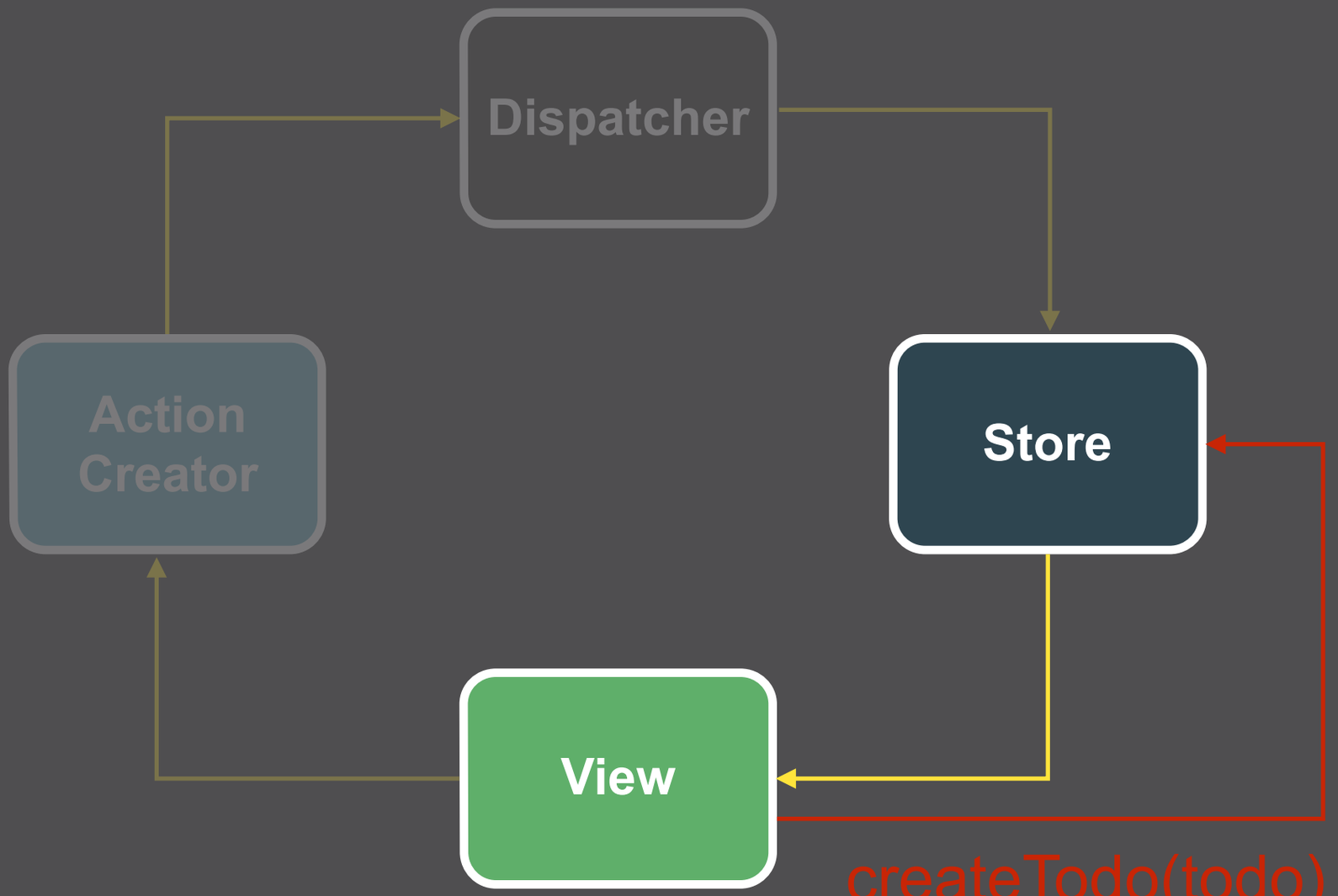
Isolering av tilstand



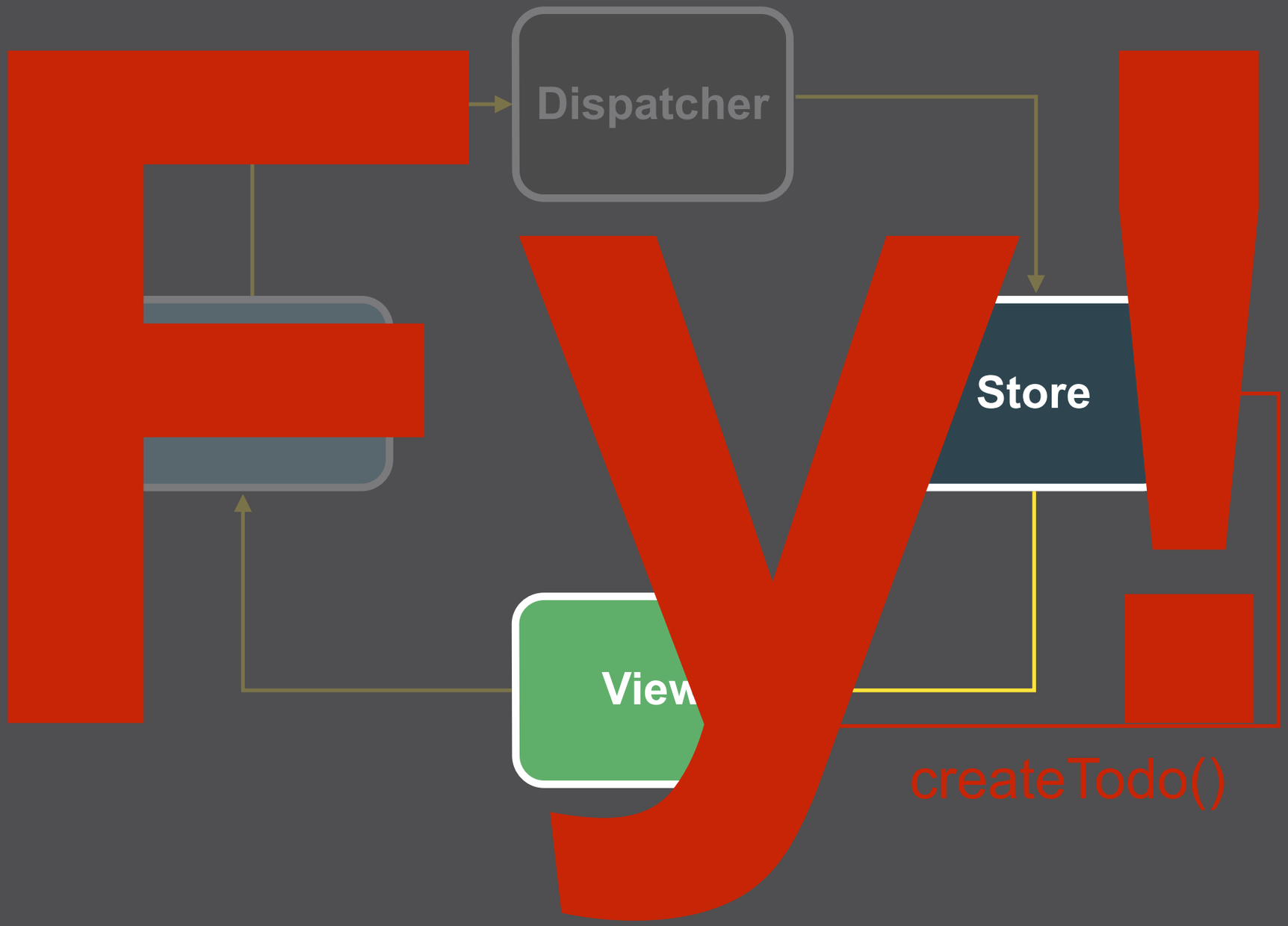
GUI-tilstand i Views



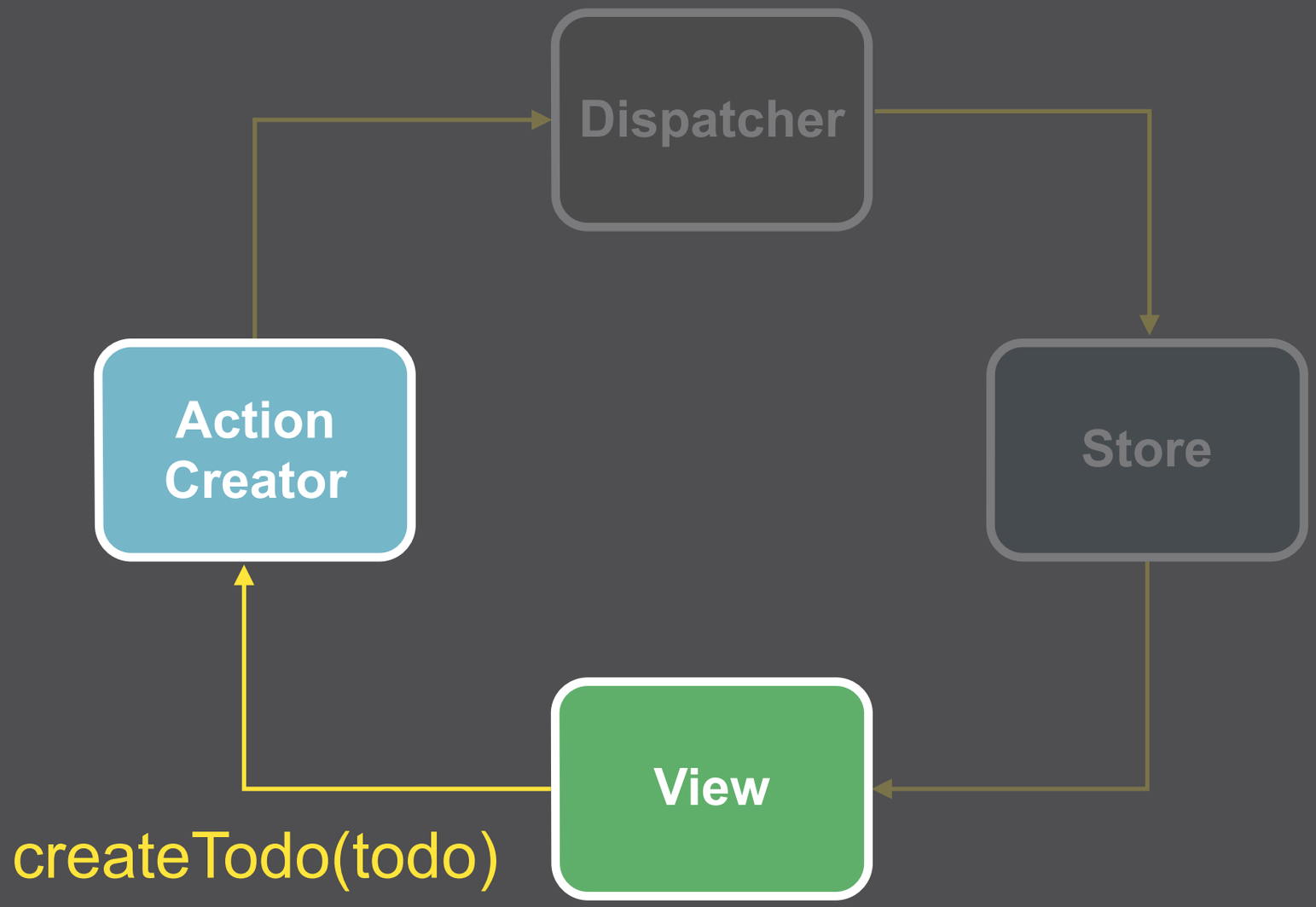
Kun lesetilgang



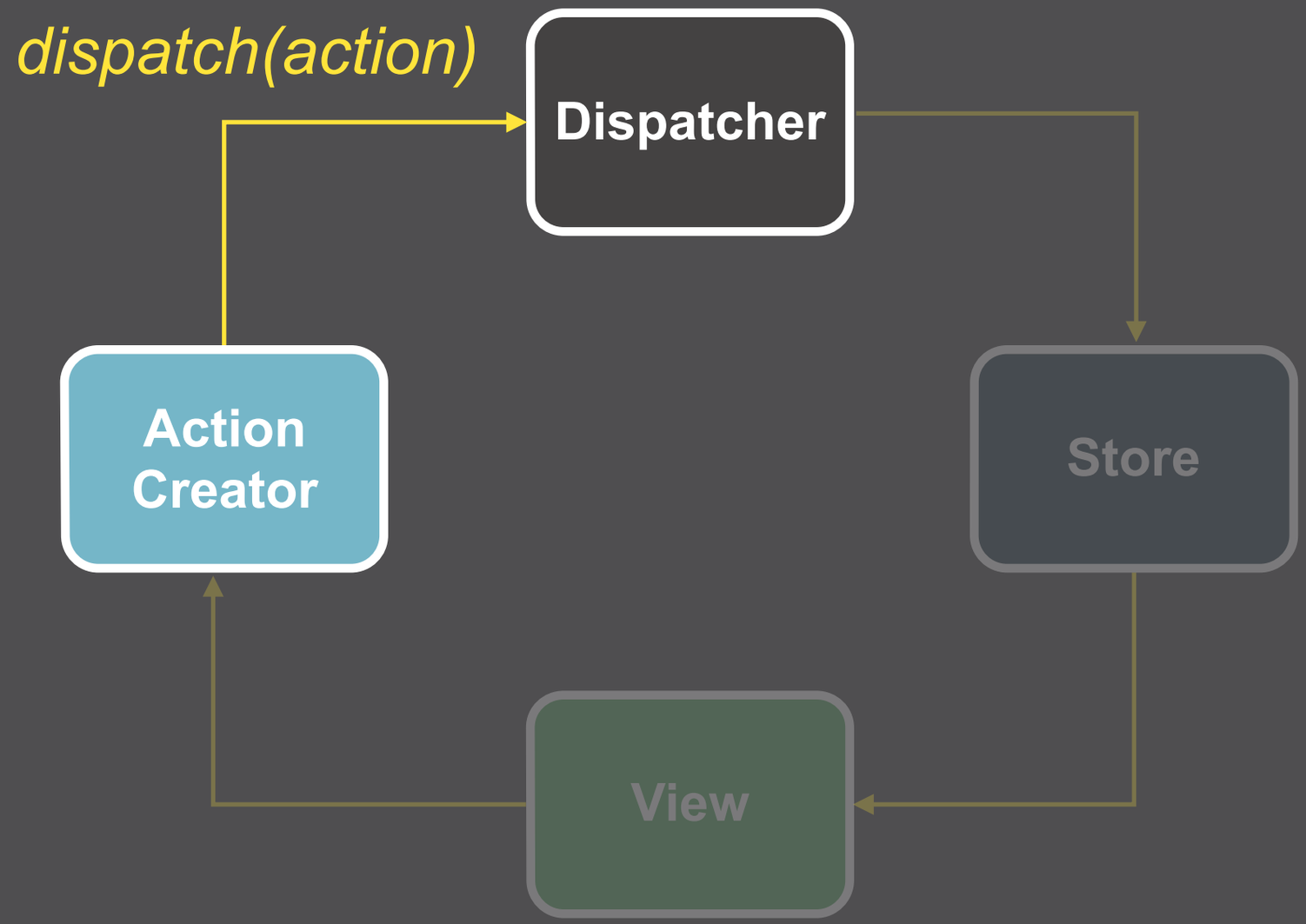
Kun lesetilgang



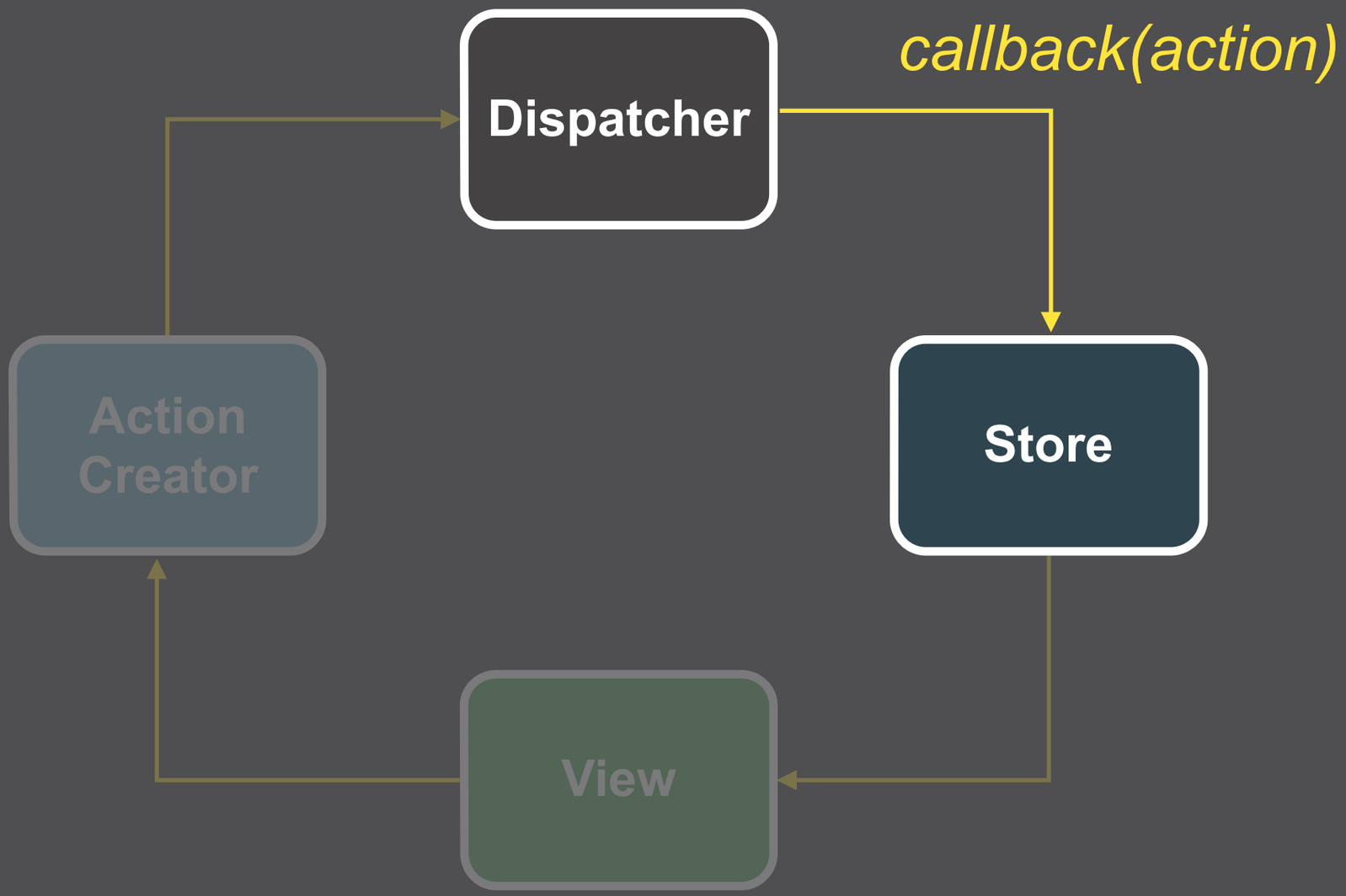
Kun lesetilgang



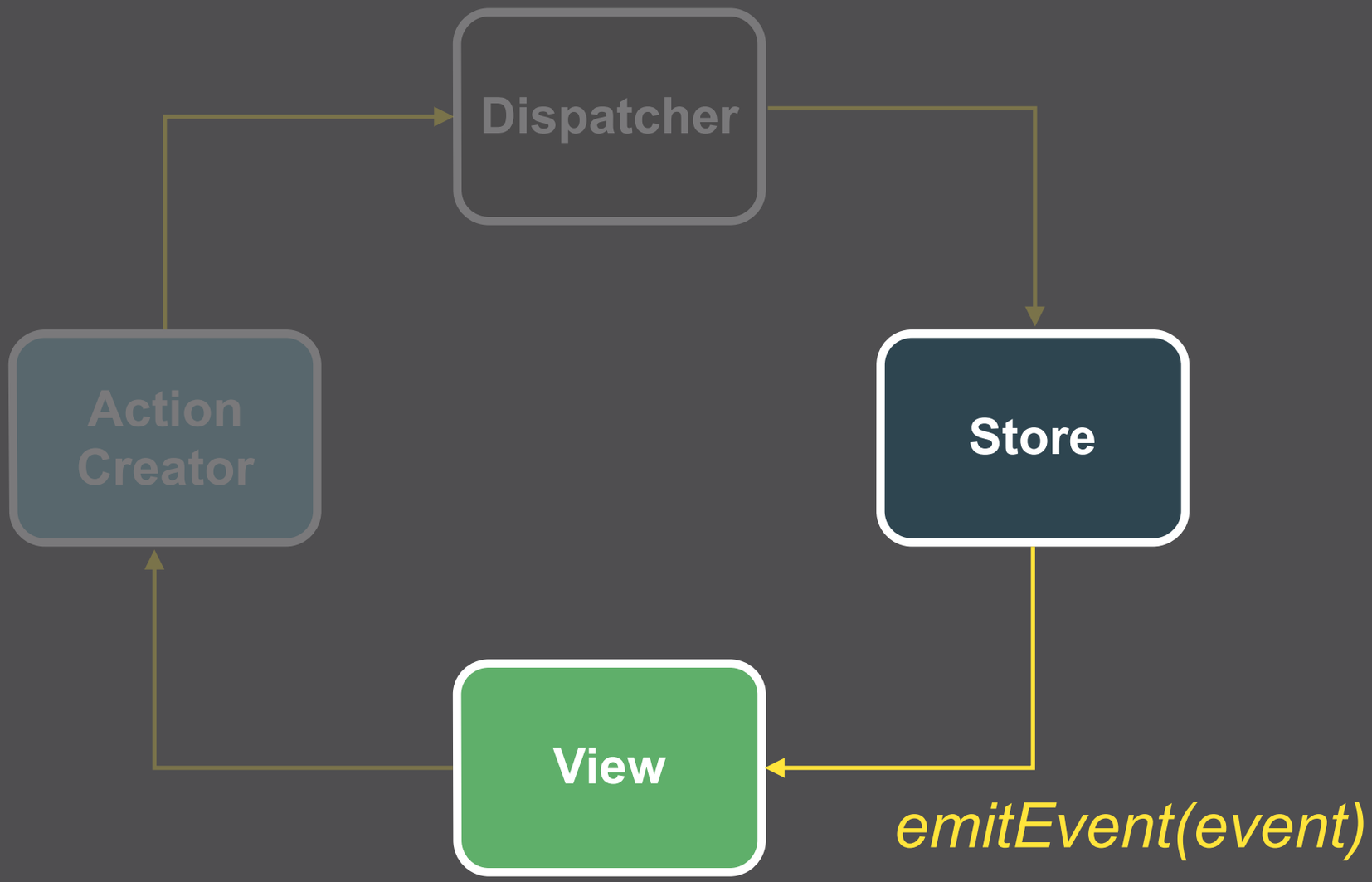
Kun lesetilgang



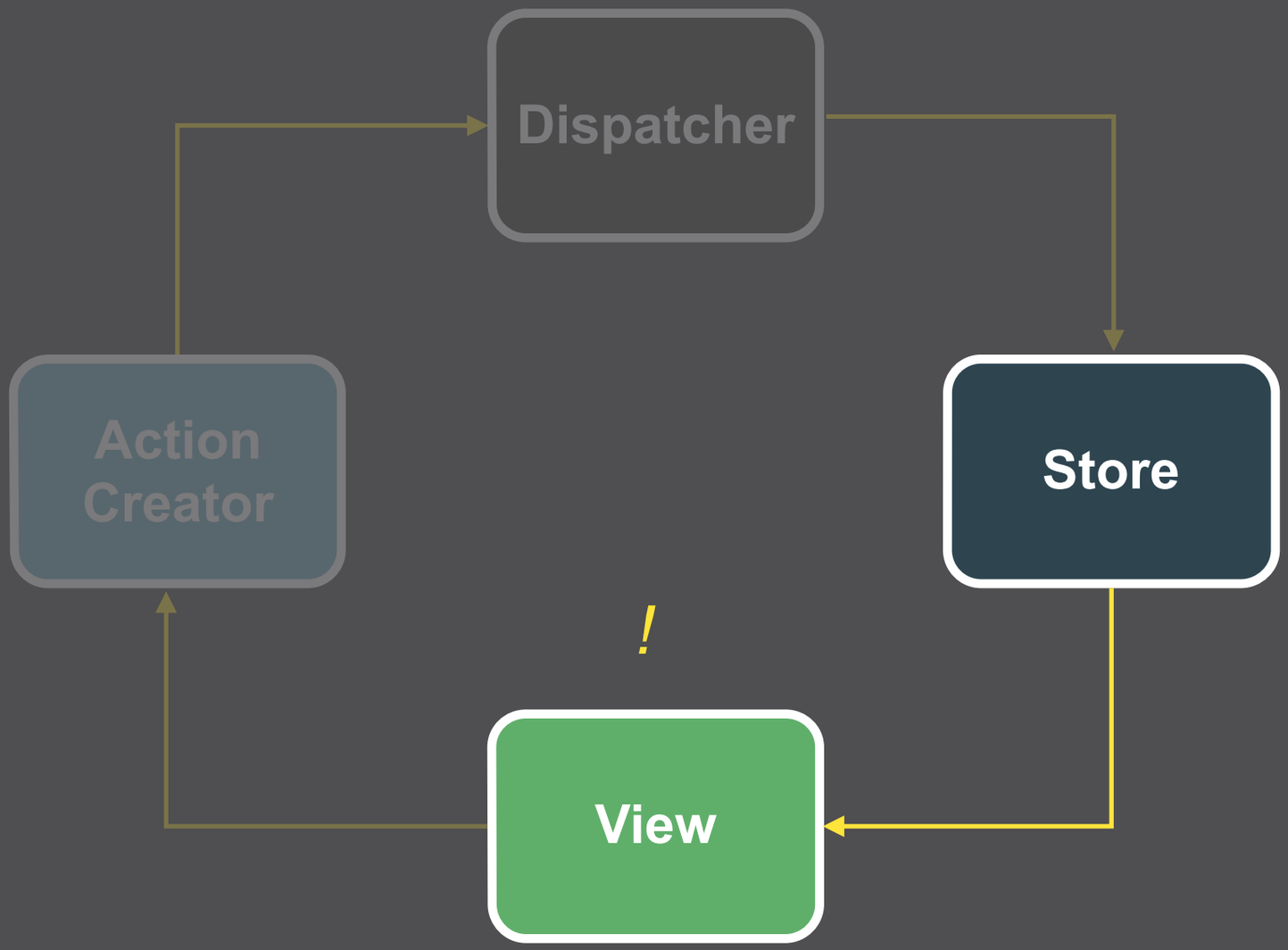
Kun lesetilgang



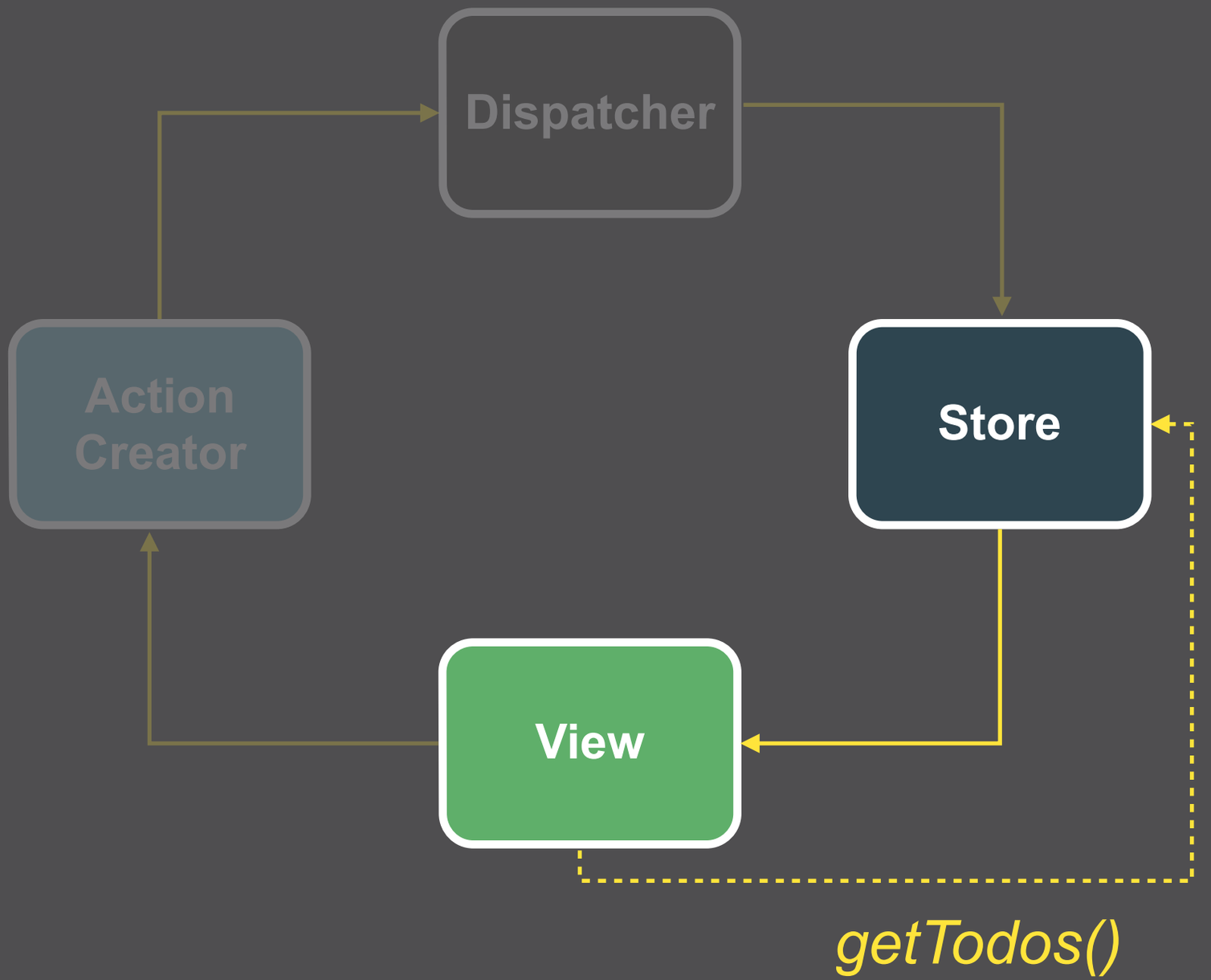
Kun lesetilgang



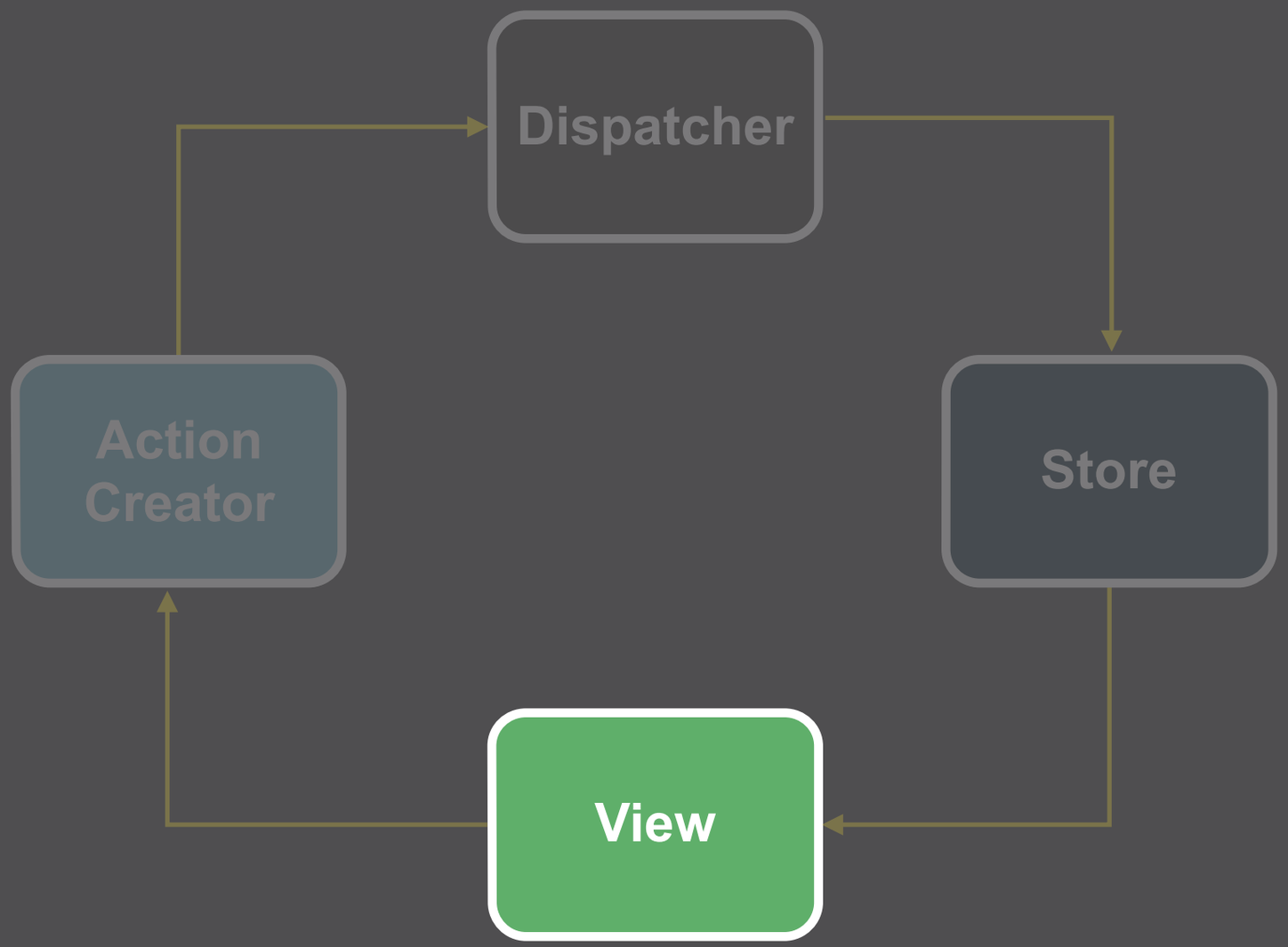
Kun lesetilgang

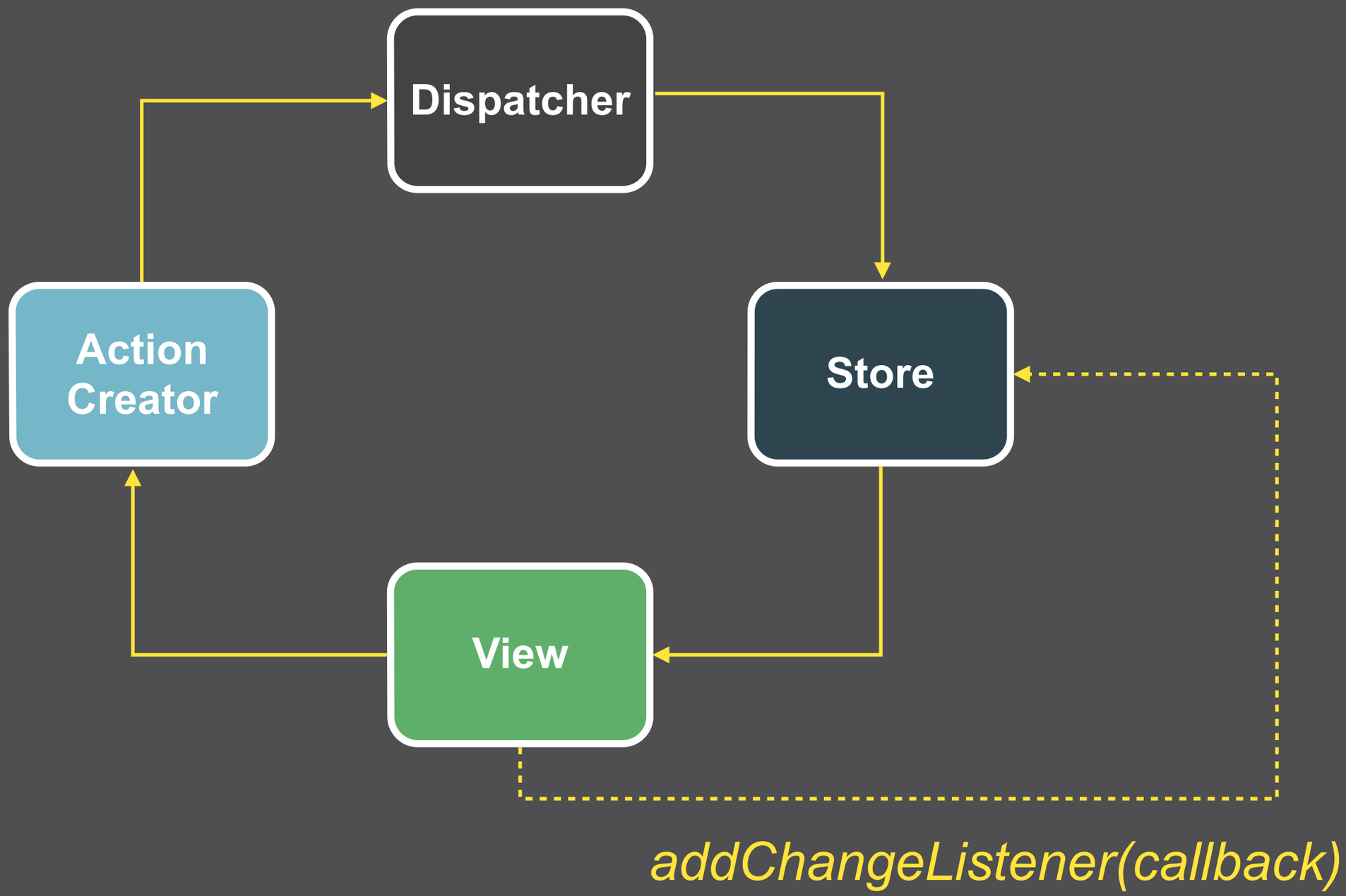


Kun lesetilgang



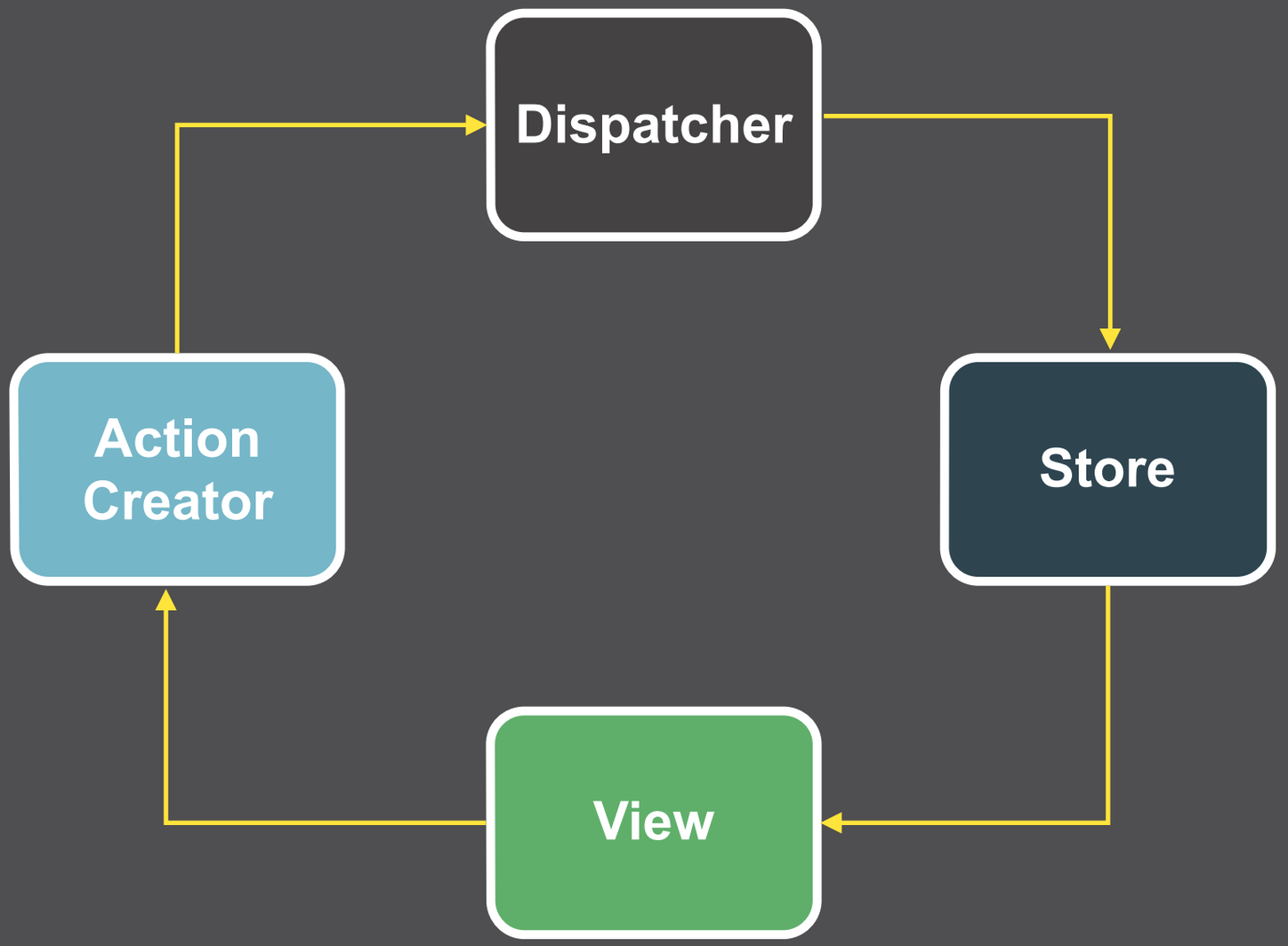
- Registrer seg hos dispatcher vha *register(callback)*
- Inneholder all forretningslogikk og tilstand
- Én store for hvert domene
- Oppdaterer views via events (EventEmitter)


Dispatcher
Action Creator
Store
View

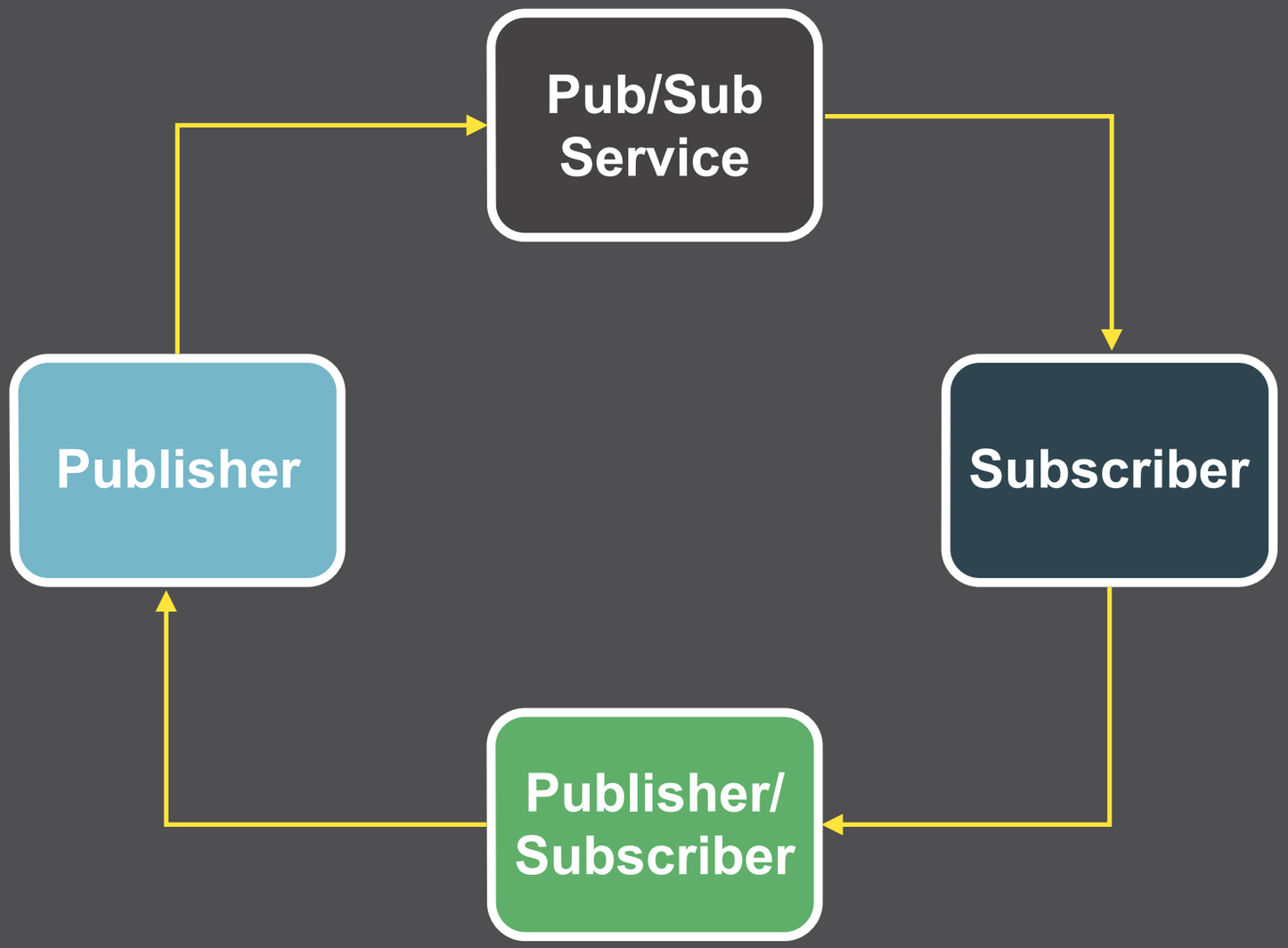

Dispatcher
Action Creator
Store
View
addChangeListener(callback)

- Flux er ment som et komplement til React
- Controller views lytter på stores
- Mottar events fra stores (Event Emitter)
- Kun lesetilgang til stores

Flux minner meg om…



Pub/Sub


Pub/Sub Service
Publisher
Subscriber
Publisher/ Subscriber

Pub/Sub

“Dispatcher is used to broadcast payloads to registered callbacks. ***This is different from generic pub-sub systems in two ways:***

1. *Callbacks are not subscribed to particular events.*
2. *Every payload is dispatched to every registered callback.*

Callbacks can be deferred in whole or part until other callbacks have been executed.” [0]

[0]: http://facebook.github.io/flux/docs/dispatcher.html

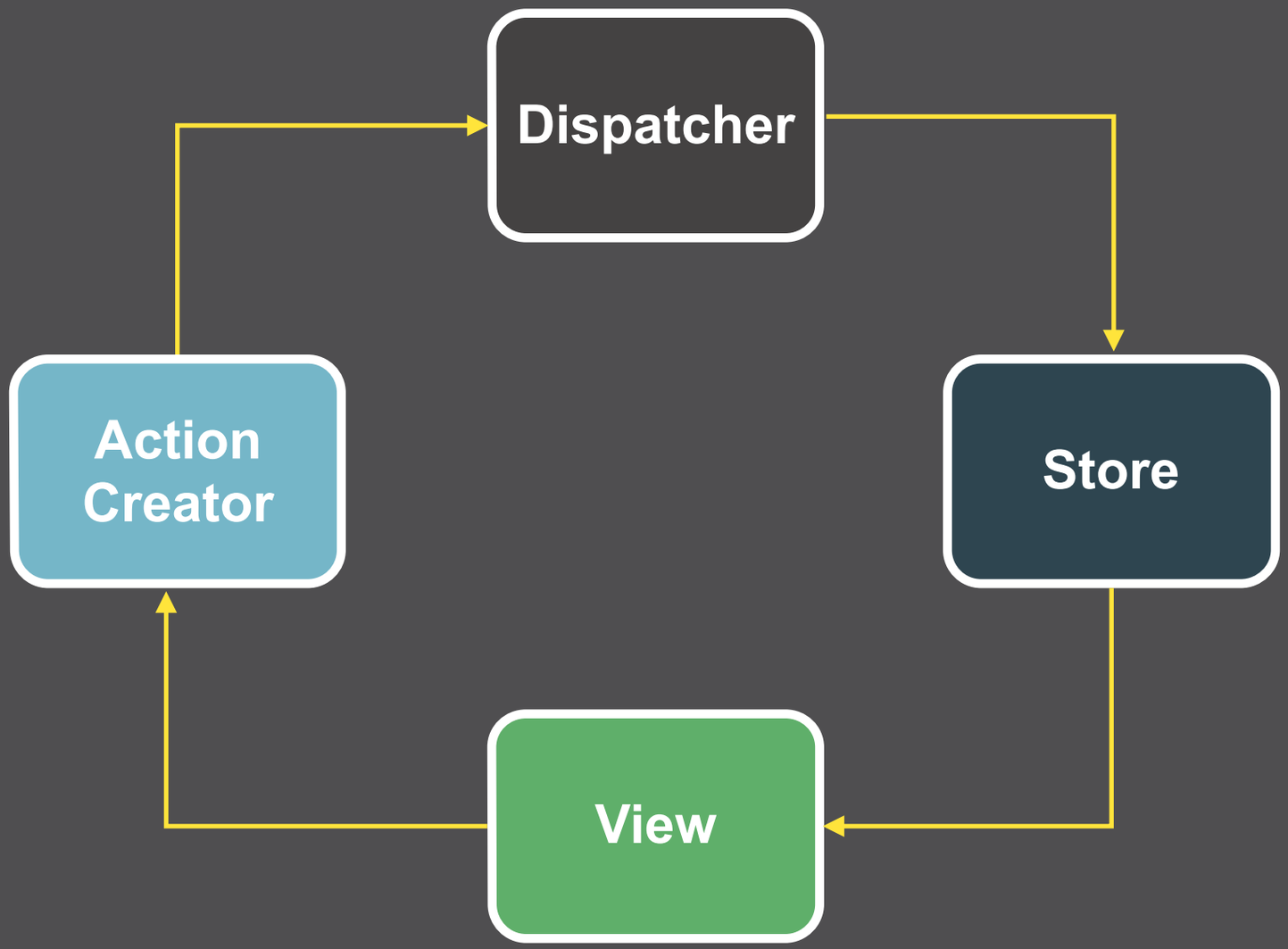
Hvorfor?

Hvorfor Flux?

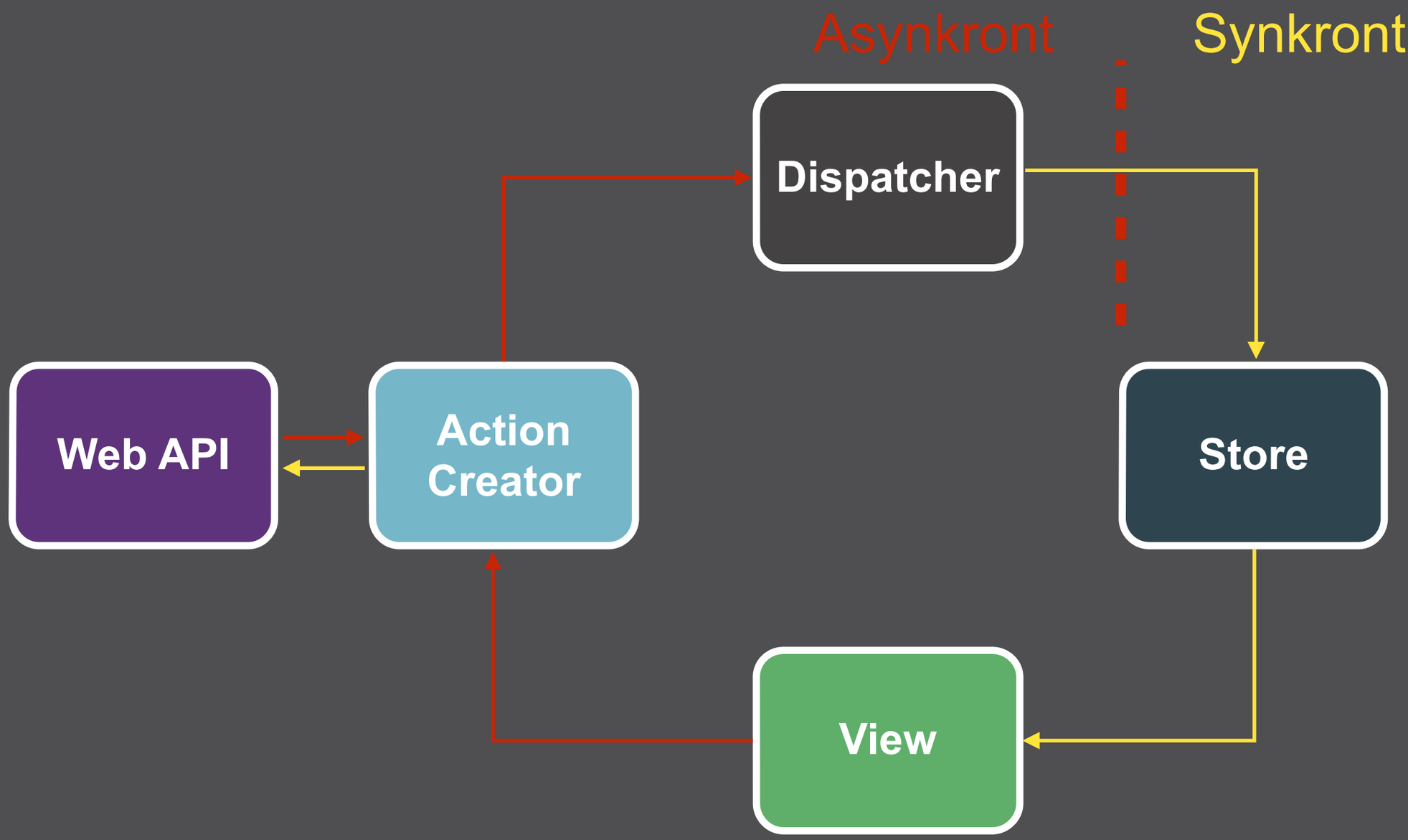
- Konseptuelt enkelt
- Forutsigbart
- Lett å resonnere rundt
- Isolerer asynkronitet og tilstand

Oppsummering

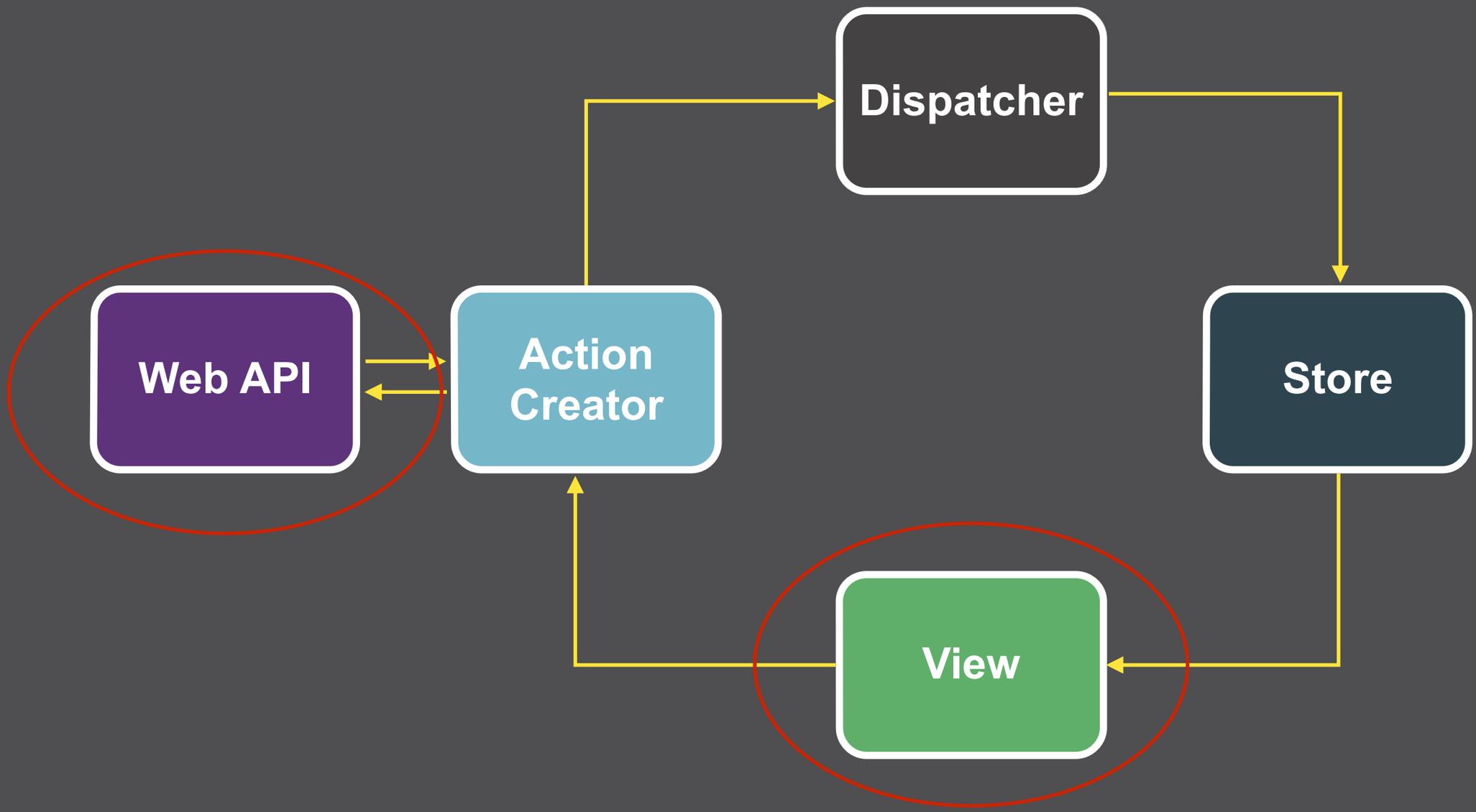
Dataflyten går kun én vei



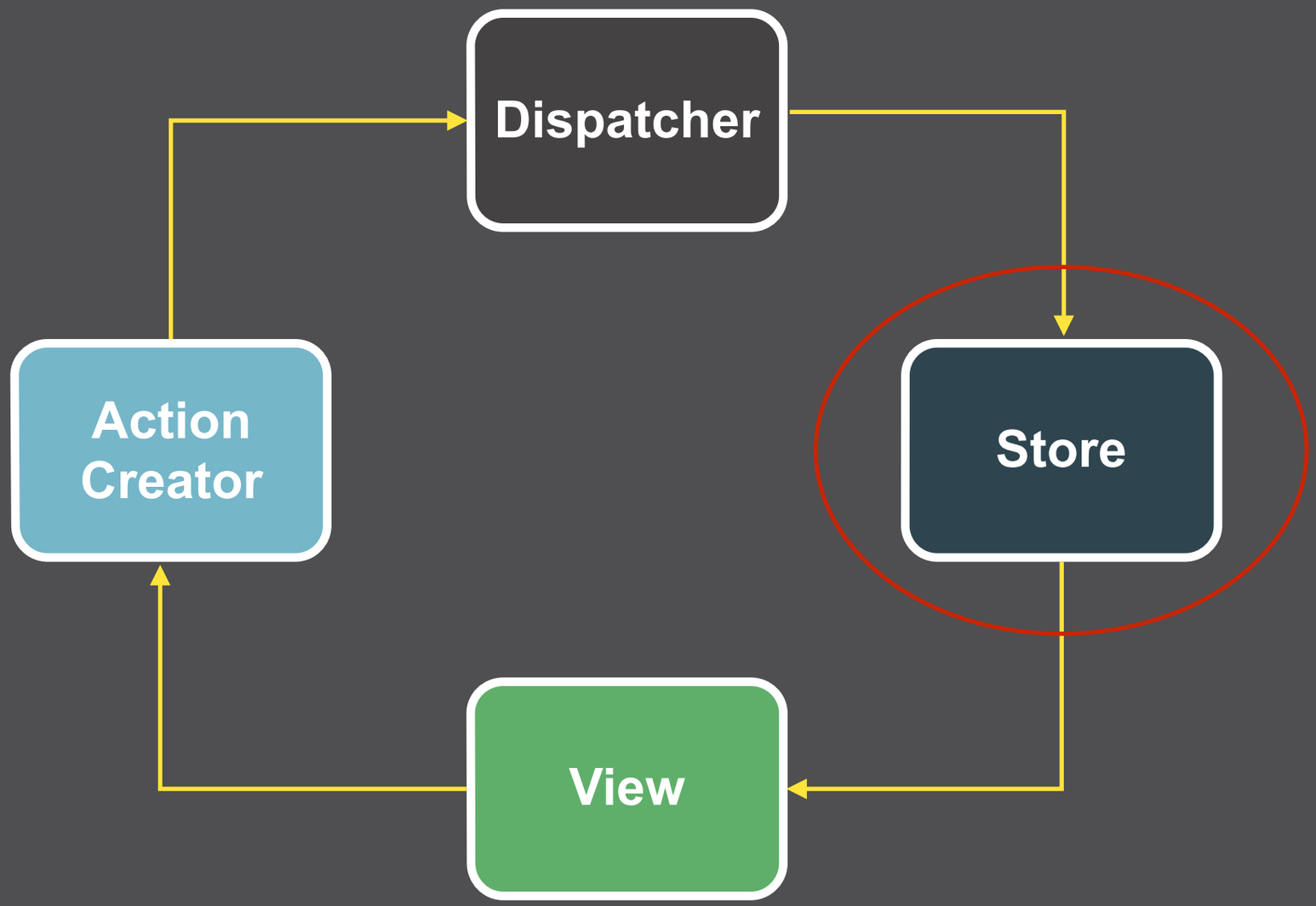
Async “barriere”



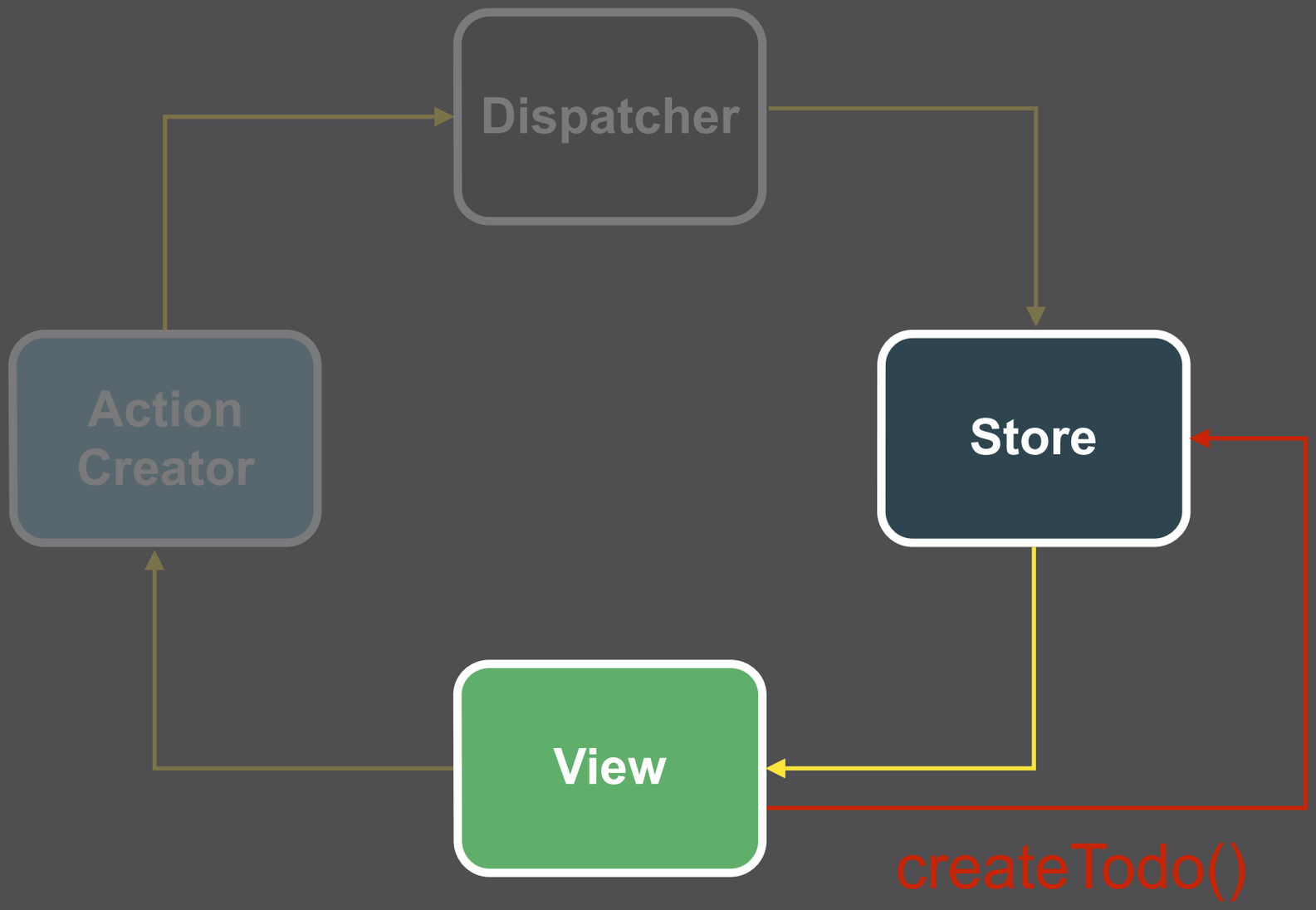
Isolering av async



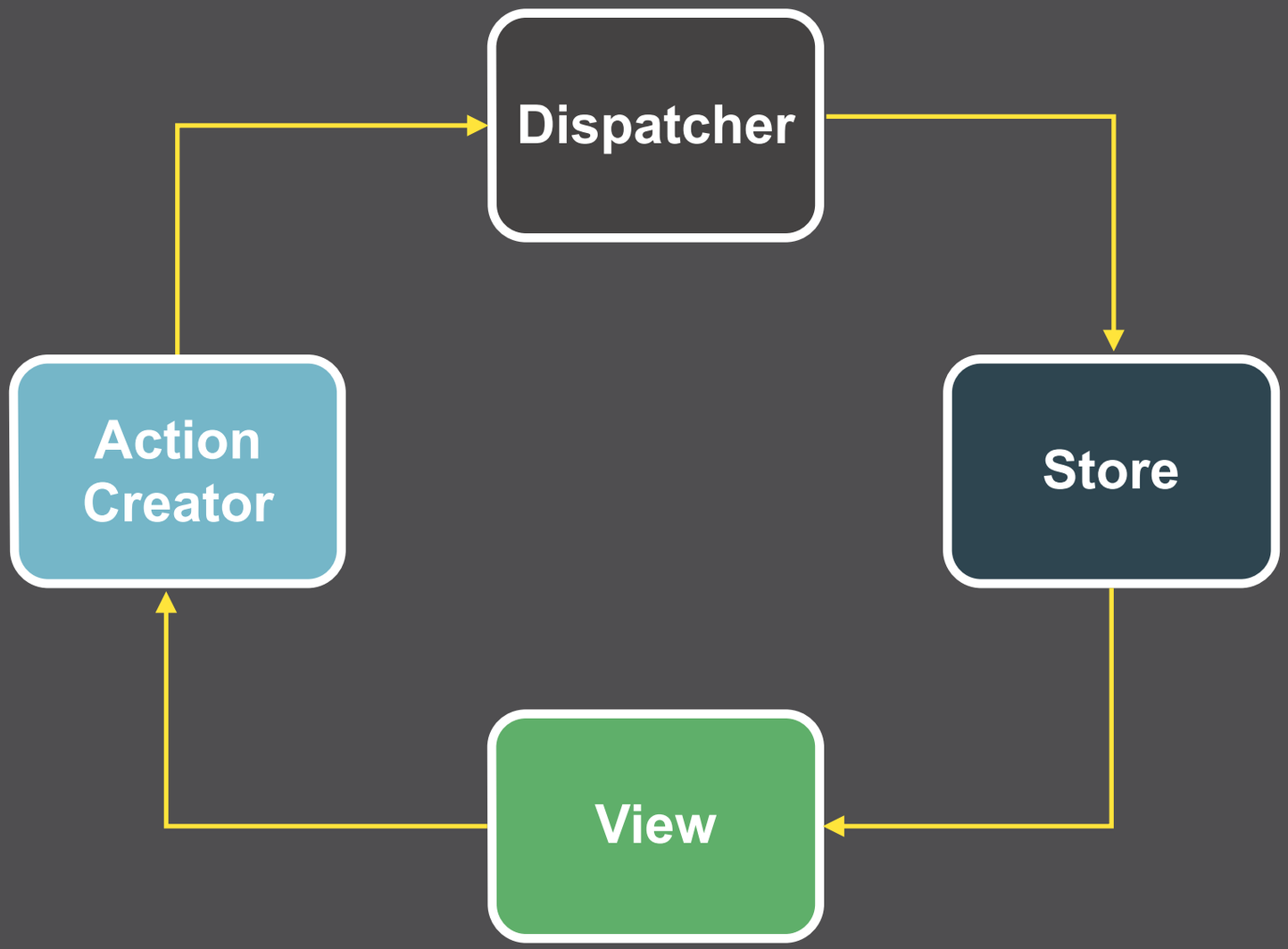
Isolering av tilstand



Kun lesetilgang



Mental modell



Framover

- Finsliping av konseptene (reflux, fluxxor, fluxible)
- Mer påvirkning fra funksjonelle paradigmer (frp, bacon.js, rxjs)
- om (Clojurescript), omniscient (JS)
- Flow, Typescript

https://github.com/kjbekkelund/flux-workshop